STADIA
HELSINGIN AMMATTIKORKEAKOULU

TEKNIIKAN JA LIIKENTEEN TOIMIALA

Sähkö- ja tietoliikennetekniikka

Ohjelmistotekniikka

INSINÖÖRITYÖ

GATEKEEPER-KÄYTTÄJIENHALLINTAOHJELMA

Työn tekijä: Markku Vitikainen
Työn valvoja: Erja Nikunen
Työn ohjaaja: Markku Valkealahti

Työ hyväksytty: __. __. 2006

Erja Nikunen
yliopettaja

STADIA
HELSINGIN AMMATTIKORKEAKOULU

TEKNIIKAN JA LIIKENTEEN TOIMIALA

INSINÖÖRITYÖN TIIVISTELMÄ

Tekijä: Markku Vitikainen

Työn nimi: GateKeeper - käyttäjienhallintaohjelma

Päivämäärä: 19.4.2006

Sivumäärä: 43

Koulutusohjelma:

Sähkö- ja tietotekniikka

Suuntautumisvaihtoehto:

Ohjelmistotekniikka

Työn valvoja: Erja Nikunen

Työn ohjaaja: Markku Valkealahti

Tämän insinöörityön tavoitteena oli suunnitella ja toteuttaa selainpohjainen käyttäjienhallintaohjelma GateKeeper Karttakeppi.com-työryhmälle. GateKeeper-ohjelma kuuluu osana laajempaan ohjelmistoprojektiin, jonka tarkoituksena on tuottaa opetusmateriaalia etäopiskelijoille Internetissä.

Etäopetusmateriaalin tuottamisessa käytetään useita ohjelmia, joille on määriteltävä eritasoisia käyttäjätunnuksia ja käyttöoikeuksia. Ohjelmien käyttäjienhallintamekanismit ovat erilaisia, mikä osaltaan lisää järjestelmän ylläpitäjien työtä ja vastuuta. GateKeeper-ohjelman tavoitteena on nopeuttaa, helpottaa ja yksinkertaistaa käyttäjienhallintaan liittyviä toimintoja tietoturvallisesti.

GateKeeper-ohjelma on suunniteltu siten, että uusien käyttäjien ja ohjelmien lisäämiselle ei ole asetettu rajoja. Jokaisen lisättävän ohjelman yhteydessä on kuitenkin selvitettävä kyseisen ohjelman käyttäjienhallintamekanismi. Tässä insinöörityössä tutkittiin, miten versionhallintaohjelma Subversionin, julkaisujärjestelmä Mambon, sekä oppimisalusta Moodlen käyttäjiä voidaan keskitetysti hallinnoida GateKeeper-ohjelman avulla.

GateKeeper-ohjelma noudattaa arkkitehtuuriltaan kolmikerrosmallia, jossa käyttöliittymä, sovelluslogiikka ja tietovarasto on erotettu toisistaan. Ohjelma koostuu joukosta PHP-kielellä toteutettuja dynaamisia web-sivuja. Tietovarastona käytetään tietokantapalvelimella sijaitsevaa MySQL-tietokantaa.

Avainsanat: käyttäjienhallinta, MySQL, Subversion, Moodle, Mambo

STADIA
HELSINGIN AMMATTIKORKEAKOULU

TEKNIIKAN JA LIIKENTEEN TOIMIALA

ABSTRACT

Name: Markku Vitikainen

Title: Creating and Managing User Accounts with GateKeeper program

Date: 19 April 2006 **Number of Pages**: 43

Department: Information engineering

Study Programme: Software engineering

Instructor: Erja Nikunen, Principal Lecturer

Supervisor: Markku Valkealahti

The objective of this graduate study was to develop a centralised management system of user accounts for three different programs that were used in one larger software project. The three programs incorporated in the system were Mambo, Moodle and Subversion.

The latest methods of software engineering were taken advantage of in this study. The whole development process was complied with the waterfall model of software engineering. This waterfall model was chosen to ensure the quality, reliability, and maintainability of the developed software.

The development process of the system was organised around the following four phases: analysis, design, implementation, and testing. The analysis was based on a requirement specification given by the client. The design phase was organised around three core workflows: planning of the module architecture, database and user interface The technical specification was written as a conclusion to the design phase. Finally, the testing plan was written on the basis of the requirement specification. The testing plan incorporates both the security testing and the system testing.

The waterfall model offers an orderly structure for software development. Close communication between a client and a developer is necessary to attain an understanding of the system requirements. Good documentation for the requirements, architecture and detailed desing are accepted methods to ensure high software quality, reliability, and maintainability of the system. These methods were found highly useful in this project.

Keywords: Software engineering, requirement specification, technical specification, three-tier model, water fall model, Mambo, Moodle, Subversion.

SISÄLLYS

TIIVISTELMÄ

ABSTRACT

SISÄLLYS

SYMBOLILUETTELO

SYMBOLILUETTELO

Evo-malli	*Evolutionary delivery,* vaihejakomalli
GateKeeper	Työnimi toteutettavalle käyttäjienhallintaohjelmalle
GLP	*GNU GLP,* GNU yleinen lisenssi
Karttakeppi.com	Laajempi ohjelmistoprojekti, jolla toteutetaan virtuaalinen oppimisympäristö
Mambo	Dynaaminen sisällönhallintaohjelma
Moodle	Ilmainen, avoimen lähdekoodin oppimisalusta
MySQL	Tiedon hallintajärjestelmä
PHP	Internet-palvelimen ohjelmointitekniikka
Repository	SVN:n tietojärjestelmä
SVN	*Subversion*, versionhallintajärjestelmä
TTY	Tampereen Teknillinen Yliopisto

1 JOHDANTO

GateKeeper-ohjelma liittyy osaltaan laajempaan ohjelmistoprojektiin, jonka tarkoituksena on tuottaa opetusmateriaalia etäopiskelijoille Internetissä. Ohjelmistoprojekti on aloitettu Tampereen Teknillisen Yliopiston Hypermedia-kurssilla viime vuonna. Kurssilla muodostettu työryhmä on suunnitellut teknisen toteutustavan, jonka avulla on mahdollista tarjota ohjattua opetusta verkkoympäristössä. Tavoitteena on kehittää loppukäyttäjälle tarjottava tuote, joka mahdollistaa opiskelun verkkoympäristössä ympäri vuorokauden. Opetusmateriaali on käytettävissä Internet-selaimella, Internet-yhteydellä varustetulla kotitietokoneella.

Karttakeppi.com on projekti, jossa hyödynnetään tietotekniikan uusimpia menetelmiä. Näillä menetelmillä pyritään tuomaan markkinoille täysin uudenlainen tuote. Valmista tuotetta hyödynnetään mahdollisesti tulevaisuudessa myös kaupallisessa toiminnassa. Uusien työvälineiden ja ohjelmien käyttö vaatii perehtymistä ja asiantuntemusta. Työryhmän vahvuutena uskotaan olevan sen jäsenten merkittävä osaaminen useilla tietotekniikan osa-alueilla. Tuotteen kehityksessä tämä voi olla ratkaiseva kilpailuetu. Myös opetusmateriaalin tuotantoon kiinnitetään erityistä huomiota. Materiaali valmistetaan itse ja materiaalin tuotantoon valitaan mahdollisimman sopivat henkilöt. Tavoitteena on laadukkaan ja toimivan tuotteen aikaansaaminen.

Karttakeppi.com-projektissa on sen alusta lähtien kiinnitetty huomiota hyvään tietoturvallisuuteen. Saavutettujen kilpailuetujen säilyttäminen edellyttää että tuotettu tieto luovutetaan vain nimetyille tahoille. Siten pyritään varmistamaan tiedon luottamuksellisuus, eheys sekä käytettävyys. Työryhmä haluaa säilyttää itsellään mahdollisuuden myöntää oikeuksia projektin tietojen käyttämiseen ja muokkaamiseen. Tietojen luvaton käyttö voisi johtaa helposti koko projektin vaarantumiseen tai kilpailuetujen menettämiseen. Viestintävirasto Ficoran mukaan tietoturvauhkien pienentämiseksi on käytettävissä useita suojautumismekanismeja. Niiden avulla tietoturvauhkien toteutuminen pyritään estämään tai uhkien vaikutusta pyritään pienentämään. Tietoturvaa lisääviä keinoja ovat esimerkiksi käyttäjien todentaminen sekä tietoliikenteen salausmekanismit. /1./

Työryhmän päätöksen mukaisesti ohjelmisto ja lopputuote toteutetaan kokonaisuudessaan vapaan lähdekoodin tuotteilla. Opetusmateriaali ja projektissa käytettävät ohjelmat on asennettu tarkoitusta varten olemassa olevalle palvelimelle. Opetusmateriaalin tuotannossa ja esityksessä käytetään tällä hetkellä ohjelmia, kuten sisällönhallintajärjestelmä

Mambo sekä kurssien hallintaohjelma Moodle. Opetusmateriaalin esittämisessä yleisin käytetty ohjelmointikieli on PHP-skriptikieli. Lisäksi projektissa on käytetty tuotetun materiaalin hallintaan tarkoitettuja versionhallintaohjelmia Subversion sekä TortoiseSVN. Projektiin käytettyjä työtunteja seurataan työtuntien kirjausjärjestelmän Dutyhoursin avulla.

Useimmat ohjelmistoprojektissa käytetyt ohjelmat on suojattu käyttäjätunnuksen ja salasanan avulla. Tämä lisää osaltaan tietoturvallisuutta, sillä ohjelmien käyttöä voidaan silloin valvoa. Järjestelmän ylläpitäjille käyttäjätunnusten hallinta asettaa kuitenkin haasteita, sillä ohjelmien omat suojausmekanismit ovat erilaisia ja yhtä käyttäjää koskevat muutokset joudutaan tekemään useisiin eri kohteisiin. Käyttäjätietojen muutosten yhteydessä järjestelmän valvojat joutuvat käytännössä tallentamaan muutokset jokaiseen ohjelmaan erikseen, ohjelmien omilla työkaluilla. Aina ei ole edes helppoa palauttaa mieleen työkaluille ominaisia piirteitä.

Ohjelmien ja käyttäjäkunnan kasvaessa käyttäjäoikeuksia koskevien muutosten tekeminen muodostuu yhä työläämmäksi. Muutosten tekeminen hidastuu ja virheiden määrä kasvaa. Tämä haittaa ylläpidon lisäksi myös käyttäjiä. Useissa ohjelmissa ei ole mekanismia historiatietojen keräämistä varten. Jälkikäteen on hankalaa nähdä kuka ja milloin muutoksia on tehnyt. Näistä syistä johtuen työryhmä päätyi keskitetyn käyttäjähallintaohjelman toteuttamiseen. Tässä tarkoituksessa syntyi GateKeeper-käyttäjienhallintaohjelmisto.

2 YLEISKATSAUS OHJELMISTOPROJEKTIIN

2.1 Ohjelmistotuotanto projektina

Tässä ohjelmistoprojektissa on pyritty hyödyntämään nykyaikaisia ohjelmistotuotannon menetelmiä ja tekniikoita. Projektin suunnittelussa tärkeimpänä ohjekirjana on ollut Ilkka Haikalan ja Jukka Märijärven Ohjelmistotuotanto-teos /2/. Tuotettujen dokumenttien tyyliin ja aihepiirien rajaukseen on otettu mallia TTY:n Menetelmät ja työohjeet -verkkodokumentista /3./

Ohjelmistotuotanto-teoksen mukaan ohjelmistojen kehittäminen jaetaan useimmiten peräkkäin tai rinnakkain eteneviin vaiheisiin. Vaiheistuksella pyritään erottamaan pienemmät osaprojektit toisistaan, jolloin kuhunkin osa-alueeseen voidaan soveltaa niihin sopivia menetelmiä ja työkaluja. Jokaiseen osaprojektiin liittyy myös toimenpiteitä, joilla pyritään varmistamaan asetettu laatutavoite. Toimenpiteet ovat yleensä tarkastuksia ja testauksia, joiden avulla havaitut virheet pystytään poistamaan. Eri vaiheissa kertynyt tieto kirjataan dokumentteihin, jotka auttavat vaiheesta toiseen siirtymistä. Edellisen vaiheen dokumentti toimii yleensä seuraavan vaiheen spesifikaationa. Spesifikaatioiden avulla on mahdollista arvioida eri vaiheiden onnistumista.

2.2 GateKeeper-ohjelman kehitysprojekti

GateKeeper-ohjelmistoprojekti on selkeä kokonaisuus, joka on osa huomattavasti laajemmasta Karttakeppi.com-projektista. Tämä insinöörityö käsittelee vain GateKeeper-ohjelmistoprojektia ja laajempaan projektiin viitataan ainoastaan tarvittaessa. GateKeeper-ohjelmistoprojekti on tämän insinöörityön tekijän opinnäytetyö, joka käsittää kokonaisuudessaan ohjelmiston suunnittelun, toteutuksen sekä dokumentoinnin.

Projekti alkoi marraskuussa, vuonna 2005. Asiakas eli Karttakeppi.com-työryhmän edustaja esitti tämän insinöörityön tekijälle idean käyttäjienhallintaohjelmasta. Hieman myöhemmin asiakas toimitti tekijälle myös toiminnallisen määrittelydokumentin, jossa oli jo kuvattu ohjelman teknisiä vaatimuksia ja ominaisuuksia /4/. Määrittelydokumentissa kerrottiin ohjelman toiminnoista, käyttöliittymästä sekä kommunikoinnista muiden ohjelmien kanssa. Tämän kaltaisia toivomuksia kutsutaan asiakasvaatimuksiksi.

Toiminnallisten vaatimusten selventämiseksi järjestettiin myös yhteinen tapaaminen Tampereella 10.12.2005.

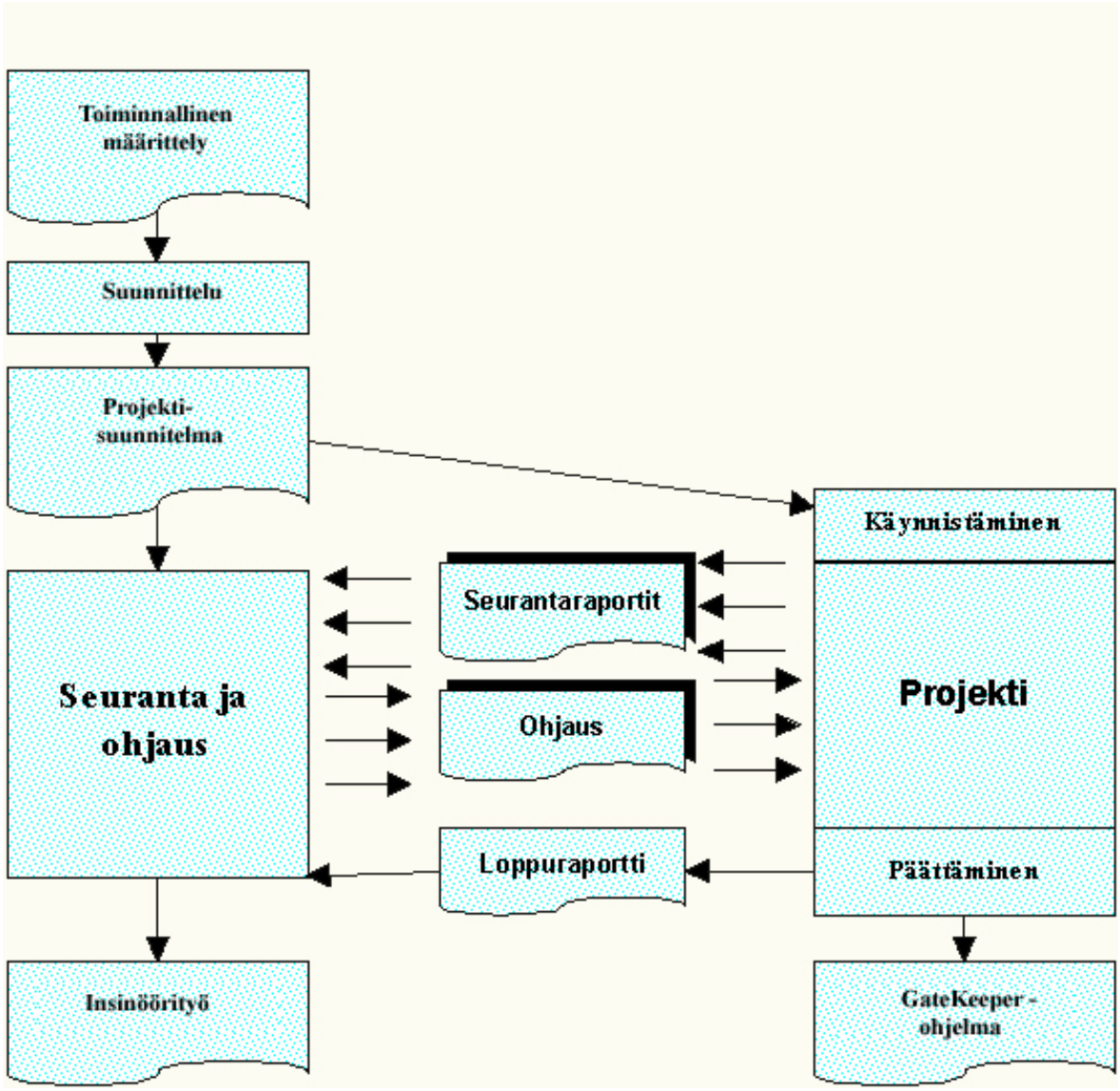
Välittömästi toiminnallisen määrittelyvaiheen jälkeen projektissa aloitettiin suunnitteluvaihe. Projektin alkuvaiheessa laadittiin projektisuunnitelma, jossa projekti jaettiin määrittely, suunnittelu-, toteutus- ja testausvaiheisiin /5/. Projektille asetettiin tavoitteet ja vaiheiden aikataulut määriteltiin. Samalla kiinnitettiin huomiota myös riskien hallintaan ja henkilöresurssien käyttöön.

Jälkikäteen voidaan sanoa, että GateKeeper-projektissa suunnittelu ja toteutusvaihe ovat edenneet lähes rinnakkain. Vaatimusmäärittelyn jälkeen aloitettiin tekninen määrittelyvaihe. Teknisessä määrittelyraportissa kuvattiin valitut ratkaisuperiaatteet, tietokantaarkkitehtuuri sekä toteutettavat moduulit /6/. Asiakkaan esittämä vaatimusmäärittely kuvasi käytettävän toteutustavan jo niin selkeästi, että ohjelman käyttöliittymän toteutus oli mahdollista aloittaa lähes heti vaatimusmäärittelyn perusteella. Teknisessä määrittelyssä eniten aikaa käytettiin tietoturvakysymysten ratkaisemiseen, sekä GateKeeper-ohjelmaan integroitavien ohjelmien käyttäjämekanismien selvittämiseen. Näistä ratkaisuista ja niiden perusteista kerrotaan tarkemmin luvussa 7. Toteutusvaiheen loppupuolella vaatimusmäärittelyn pohjalta laadittiin testaussuunnitelma /7/. Moduuli- ja integrointitestaus liittyi projektin loppuvaiheessa kiinteästi toteutusvaiheeseen. Projektin viimeisinä vaiheina suoritettiin järjestelmätestaus ja siinä havaittujen puutteiden korjaukset.

Tässä projektissa voidaan nähdä kaksi haaraa, varsinainen tuotekehitys, sekä projektiin liittyvä seuranta. Molemmilla haaroilla voidaan nimetä niihin läheisesti liittyvät sidosryhmät. Seurannasta ja ohjauksesta vastaa Helsingin ammattikorkeakoulu ja valmis tuote päätyy Karttakeppi.com -työryhmän käyttöön. Onnistuneen projektin tuloksena syntyy siis valmis tuote eli GateKeeper-ohjelma, sekä koko projektin vaiheista kertova päätösraportti eli tämä insinöörityö.

On vaikea sanoa, onko projektissa käytetty elinkaarimalli lähempänä vesiputousmallia *(waterfall model)*, vai Evo-mallia *(evolutionary delivery)*. Projektin ensimmäisen toteutusvaiheen loppupuolella näyttäisi siltä, että projekti on noudattanut vesiputousmallia. Jos projekti kuitenkin jatkuu, se saattaa saada enemmän piirteitä Evo-mallista. Kuva 1 esittää erittäin hyvin GateKeeper-projektin etenemistä. Projekti on aloitettu asiakkaan laatiman toiminnallisen määrittelyn pohjalta. Projektisuunnitelmassa on kuvattu projektiin liittyvät vaiheet ja sen aikataulu. Projektin edetessä suunnittelun jälkeen syntynyttä teknistä mää-

rittelyraporttia on korjattu ja täydennetty toteutusvaiheen edetessä. Projektin jokaiseen vaiheeseen on liittynyt ohjausta ja seurantaa sekä asiakkaan että oppilaitoksen osalta. Tavoitteena on, että lopputuotteena syntyvä GateKeeper-ohjelmisto sekä koko ohjelmistoprojektia kuvaava insinöörityö olisi projektin päätösvaiheessa tarkastettu ja hyväksytty kaikkien sidosryhmien taholta. Siten GateKeeper-ohjelma olisi myös mahdollista luovuttaa asiakkaan käyttöön heti tämän projektin päättyessä.



Kuva 1. Projektin kulku /8/

3 ASETETUT OHJELMISTOVAATIMUKSET

Asiakkaan laatiman toiminnallisen määrittelyn perusteella asiakasvaatimukset voidaan jakaa toiminnallisiin ja ei-toiminnallisiin vaatimuksiin /2,s.39/. Toiminnallisten vaatimuksien mukaan asiakas esitti, että uudella GateKeeper-ohjelmalla olisi mahdollista hallinnoida keskitetysti Karttakeppi.com-projektissa käytettäviä ohjelmia /4/. Hallinnoitavat ohjelmat ovat asiakas-palvelin -tyyppisiä ohjelmia, joissa käytetään yhteisen palvelimen resursseja. Tästä syystä asiakkaalla on tarve valvoa ja tarvittaessa rajoittaa käyttäjien oikeuksia. Toisin sanoen, oikeuden palvelimen resurssien käyttämiseen saavat vain nimetyt henkilöt.

3.1 Toiminnalliset asiakasvaatimukset

Tässä luvussa referoidaan asiakkaan laatimia toiminnallisia vaatimuksia GateKeeper-ohjelmalle. /4./

GateKeeper on ohjelma, joka toimii Internet-selaimella Internetissä. GateKeeper ohjelman verkko-osoitteeseen siirryttäessä käyttäjälle avataan kirjautumissivu, mikäli käyttäjällä ei ole voimassa aktiivista istuntoa tai istunto on jo vanhentunut. Kirjautumissivulla on kaksi syötekenttää joista toinen on käyttäjätunnusta varten ja toinen salasanaa varten. Salasanalle varattu kenttä ei saa näyttää selväkielistä tekstiä näytöllä. Lisäksi sivulla on ”kirjaudu”-nappi, jota painamalla ohjelma tarkistaa, onko tunnus oikein. Jos kirjautuminen onnistuu, siirtyy ohjelma pääsivulle. Pääsivulla tulee näyttää sisäänkirjautuneen käyttäjän tunnus, sekä linkit ohjelman muihin toimintoihin.

Uuden käyttäjän lisäyksen yhteydessä käyttäjälle luodaan haluttu, yksilöllinen käyttäjätunnus. Ohjelman on tarkistettava, ettei samoja käyttäjätunnuksia voida antaa. Sivulla pyydetään uusi salasana kahdessa eri syötekentässä. Kun käyttäjä painaa näytöllä olevaa ”tallenna”-nappia, tarkistaa järjestelmä syötekenttien samanlaisuuden. Jos kenttien syötteet ovat yhtenevät, päivitetään uusi salasana kaikkiin palveluihin, joihin käyttäjä on kirjattu kuuluvaksi. Salasana tallennetaan muistiin salattuna. Tiedot tallennetaan GateKeeper-ohjelman omaan tietokantaan. Kun käyttäjä lisätään tietokantaan, hänellä ei ole oletuksena annettu oikeuksia mihinkään integroituun ohjelmaan.

Ohjelmalla on kahden tyyppisiä käyttäjiä. Toteutettavan ohjelman ylläpito-roolin käyttäjinä tulevat olemaan Karttakeppi.com-palvelun ylläpitäjät. He pääsevät muokkaamaan täysin vapaasti eri käyttäjätunnuksia palvelimella. Yksittäinen käyttäjä, jolla ei ole ylläpito-roolia, pystyy muuttamaan omaa salasanaansa, mutta ei mitään muita tietoja. Käyttäjän kirjautuessa ohjelmaan, hänelle kuuluva rooli tarkistetaan tietokannasta.

Käyttäjän tunnusta poistettaessa ei tunnusta poisteta tietokannasta vaan salasana vaihdetaan satunnaiseksi ja sulkupäivämäärä tallennetaan. Poistettuja käyttäjiä ei enää näytetä oikeuksienhallintatoiminnoissa.

Oikeuksienhallintaikkunassa jokainen eri ohjelma näkyy omana osionaan. Jokaisen ohjelman kohdalla voidaan erikseen nähdä, onko ohjelman käyttöoikeus myönnetty käyttäjälle. Uusien ohjelmien tulee olla helposti lisättävissä listaan. Ensimmäisessä vaiheessa listaan liitettävät ohjelmat ovat Mambo, Moodle ja Subversion.

3.2 Ei-toiminnalliset asiakasvaatimukset

Ei-toiminnallisilla asiakasvaatimuksilla tarkoitetaan ohjelman käyttöön vaikuttavia tekijöitä, kuten ohjelman suoritusteho, vasteaika tai käytettävyys /2, s.39/. Myös näihin tekijöihin on kiinnitettävä huomiota jo ohjelman suunnitteluvaiheessa. Kuten tiedetään, varhaisessa vaiheessa huomioitujen asiakasvaatimusten toteuttaminen on helpompaa kuin korjausten tekeminen jälkikäteen. Asiakas oli nimennyt myös ei-toiminnallisia järjestelmävaatimuksia määrittelyraporttiin.

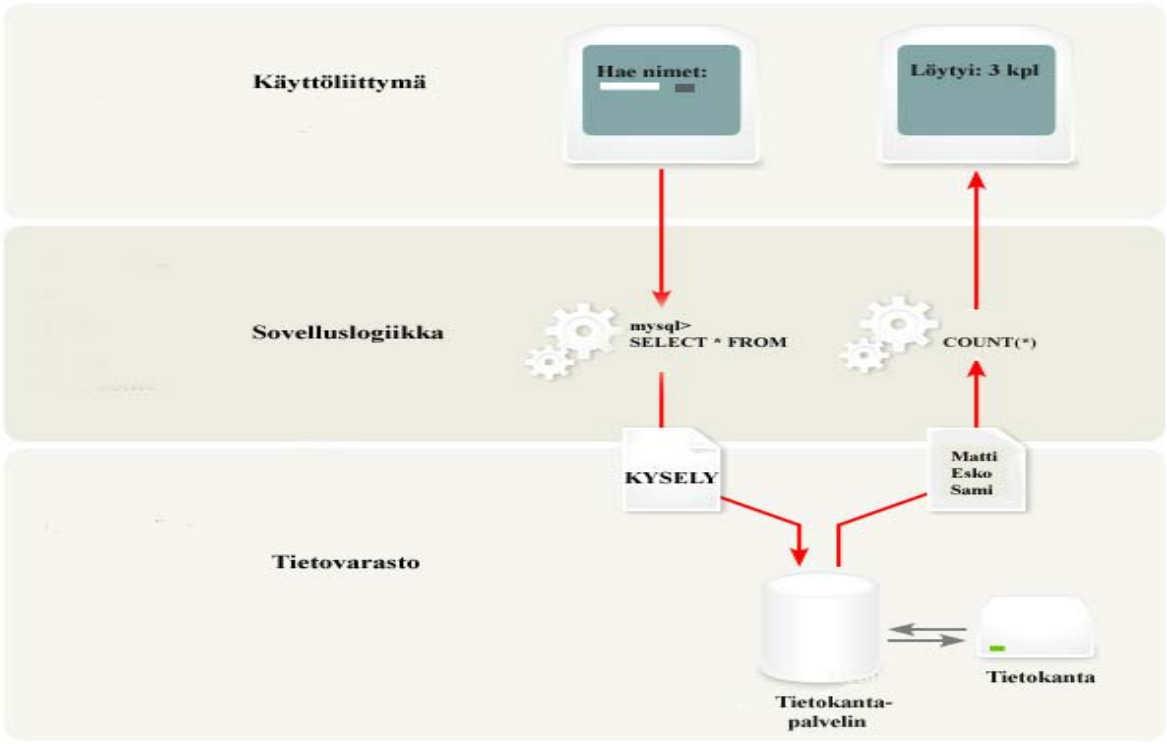
Nimettyjen järjestelmävaatimusten mukaisesti ohjelmassa ei saa olla rajoitusta lisättävien käyttäjätunnuksien määrän suhteen. Myöskään hallinnoitavien ohjelmien määrää ei saa rajoittaa. GateKeeper-ohjelman tulee tallentaa kaikki tiedot tarkoitusta varten luotuun MySQL-tietokantaan. Ohjelman tulee myös kommunikoida luotettavasti muiden projektiin kuuluvien ohjelmien kanssa. Parhaimmillaan ohjelmalla voi olla muutamia samanaikaisia käyttäjiä. Asiakkaan toivomuksen mukaisesti ohjelman pitäisi suoriutua kaikista palvelupyynnöistä alle neljässä sekunnissa. Ehdottomana rajoituksena asiakas nimesi sen, ettei ohjelmaa voisi käyttää ilman voimassaolevaa käyttäjätunnusta ja salasanaa. Tämä tarkoittaa, että ohjelmiston sivustoille sallitaan pääsy ainoastaan kirjautumissivun kautta.

4 JÄRJESTELMÄN SUUNNITTELU

Järjestelmän suunnittelun tuloksena syntyi tekninen määrittelyraportti. Tässä luvussa käsitellään teknisessä määrittelyraporttissa kuvattua järjestelmän arkkitehtuuria, ympäristöä ja ratkaisuperiaatteita. /6./

4.1 Verkkopalveluarkkitehtuuri

Toteutettavalle ohjelmalle asetettiin vaatimukseksi, että se toimisi verkkoympäristössä, Internet-selaimella. Internet on asiakas-palvelin-järjestelmä *(client/server system)*, jossa palveluja käytetään web-selaimen avulla /9, s.12/. Tietokantapohjaisen verkkopalvelun arkkitehtuuri noudattaa yleensä niin sanottua kolmikerrosmallia *(three-tier model)*, jonka periaate on esitetty kuvassa 2 /10/. Malli toimii niin, että käyttäjä käyttää palvelua asiakaskoneen avulla. Välikerroksessa toimii järjestelmän sovelluslogiikka, jonka avulla tieto noudetaan, tai tallennetaan tietovarastoon. Kerrosten välinen tiedonsiirto tapahtuu aina kerrosten välillä, eli käyttöliittymäkerros ei koskaan kommunikoi suoraan tietovaraston kanssa. Välikerros toimii siten asiakkaana tietokantapalvelimelle ja palvelimena web-selaimelle. Yleensä suurin osa sovelluslogiikasta sijaitsee palvelimella. Välikerroksen toiminnallisuus voidaan toteuttaa useilla eri ohjelmointikielillä. /11./



Kuva 2. Kolmikerrosmalli /10/

GateKeeper-ohjelman toiminta perustuu selkeästi kolmikerrosmalliin. Vaihtoehtoja tälle valinnalle oli vaikea esittää. Ohjelma tarvitsee tietovaraston, jossa käyttäjätietoja säilytetään. Ohjelman on pystyttävä kommunikoimaan myös siihen liitettyjen ohjelmien ja näiden tietovarastojen kanssa. On varmasti etu, vaikka ei välttämätöntä, että kaikki tietovarastot sijaitsevat samalla tietokantapalvelimella. Siten vältetään mahdolliset ongelmat tietoliikenneyhteyksissä. Näillä perusteilla GateKeeper-ohjelman tietovarasto päätettiin sijoittaa Karttakeppi.comin tietokantapalvelimelle. Projektin alkaessa palvelimeen oli jo asennettu Debian GNU-käyttöjärjestelmä sekä MySQL-tietokannanhallintajärjestelmä.

4.2 Ohjelmointikielen valinta

Dynaamisten web-palvelujen toteuttamiseksi on olemassa useita vaihtoehtoja /9, s.2/. Sopivat työkalut voidaan poimia monien tekniikoiden ja ohjelmointikielien joukosta. Yksi periaatteellinen ero tekniikoiden välillä on, halutaanko ohjelma suorittaa palvelimella vai asiakaskoneella. Aikaisemmin palvelimella ajettavat ohjelmat olivat yleisempiä. Palvelin suoritti ohjelman ja lähetti tuloksen asiakkaalle. Tämän kaltaisia ohjelmia kutsutaan CGI-ohjelmiksi *(Common Gateway Interface)*. Runsaan käyttäjämäärän kuormittaessa palvelinta CGI-ohjelman toiminta voi muuttua hitaaksi. Tähän ongelmaan ratkaisun tarjoaa esimerkiksi Java-ohjelmointikieli. Sillä kirjoitettu ohjelmakoodi noudetaan palvelimelta, mutta suoritetaan asiakaskoneessa. Tällöin puhutaan asiakaspuolen tietojenkäsittelystä (*client-side computing)*. Java on nykyään suosittu, järjestelmäriippumaton oliopohjainen ohjelmointikieli.

Sekä palvelin- että asiakaspuolen ohjelmilla on toisiinsa verrattuna omat etunsa sekä haittansa. CGI-ohjelmoinnin etuna on yksinkertainen tapa välittää tietoa asiakkaan ja palvelimen välillä. CGI-rajapinta ei ole ohjelmointikieli. Se sallii skriptien kirjoittamisen monilla eri kielillä. Skripti on tietokoneohjelma, jonka tarkoitus on olla helposti omaksuttavissa ja suhteellisen helppolukuinen. Tietoturvallisuuden kannalta erittäin suuri etu on, että CGI-ohjelman eli skriptin lähdekoodi pysyy käyttäjälle näkymättömänä. CGI-skriptien avulla voidaan helposti toteuttaa haku- ja tilauslomakkeita silloin, kun tieto sijaitsee palvelimella. Todellista interaktiivisuutta skripteillä ei kuitenkaan saavuteta, sillä niiden avulla toteutetut web-palvelut muodostuvat pohjimmiltaan dynaamisesti luoduista, mutta muuten staattisista HTML-sivuista /12/.

Mikäli web-palvelu halutaan tarjota CGI-rajapinnan avulla, voidaan ohjelma kirjoittaa millä tahansa ohjelmointikielellä, joka tuottaa komentoriviltä suoritettavissa olevan objektitiedoston. Yleisimmin käytettyjä kieliä ovat Perl, ASP, PHP, C/C++, Pascal tai Unixin komentokielet. Käytännössä palvelimen ylläpitäjä voi määritellä mitä CGI-ohjelmia palvelimella saadaan suorittaa.

PHP *(Hypertext Preprocessor)* on yleiskäyttöinen skriptikieli, joka sopii erityisesti web-palvelujen tuottamiseen. /13, s.17/. Se on alun perin kokoelma web-pohjaisten sovellusten tekemistä helpottavia rutiineja. PHP-ohjelmakoodi voidaan sijoittaa mihin tahansa kohtaan HTML-koodia. PHP-koodi suoritetaan palvelimella joka kerta, kun web-sivu lähetetään asiakaskoneen selaimelle. PHP on ohjelmointikieli, jonka avulla monimutkaistenkin sovellusten toteuttaminen palvelimella on mahdollista. PHP:n valmiit funktiot tarjoavat tuen useille tietokannanhallintajärjestelmille, myös luvussa 4.5 esitellylle MySQL:lle.

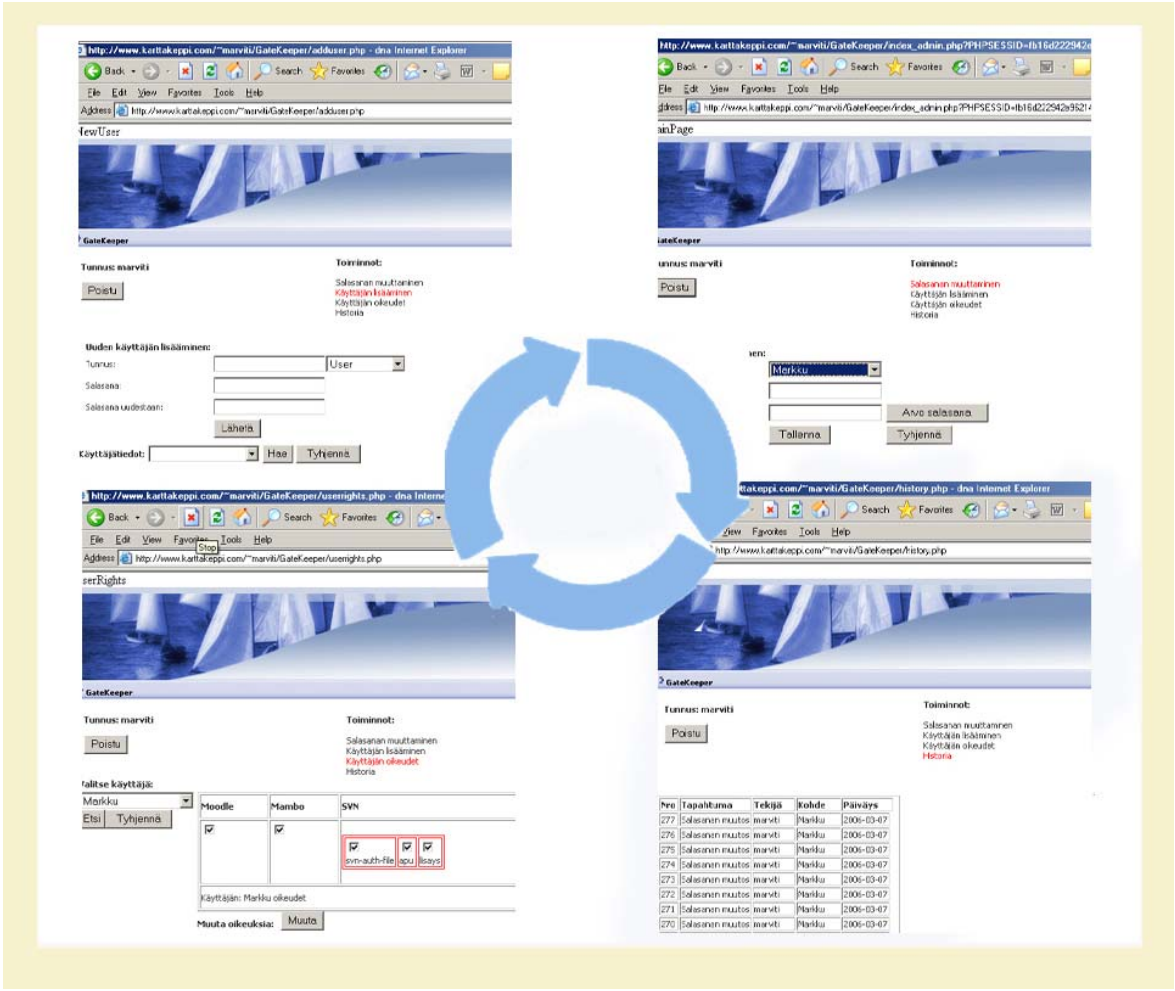
GateKeeper-ohjelmointiprojektissa tietoturvallisuuden merkitys on suuri. Valmis GateKeeper-ohjelma sijoitetaan Karttakeppi.comin palvelimelle. Koska palvelimella on jo valmius suorittaa PHP-ohjelmia, PHP-ohjelmointikielen valinta GateKeeper-ohjelmistoprojektin työvälineeksi oli luonteva. Mahdollista palvelimen hidastumista käyttäjien määrän kasvaessa ei pidetty uhkatekijänä.

4.3 Käyttöliittymä

Lähtökohta GateKeeper-ohjelmiston suunnittelulle oli, että ohjelma muodostuisi joukosta toisiinsa linkitettyjä, dynaamisia web-sivuja. Web-sivujen määrää ei rajoitettu ennalta. Sivuille olisi päästävä ainoastaan kirjautumissivun kautta, kuten toiminnallisessa määrittelyssä oli kuvattu. Ohjelman toiminnallisuutta sisältävien sivujen ulkoasusta saa käsityksen kuvasta 3.

Verkkosivuista koostuvan ohjelman rakenteeksi on olemassa periaatteessa kolme vaihtoehtoa. Lineaarisessa rakennemallissa sivulla olevat linkit viittaavat sekä seuraavaan että edelliseen sivuun. Hierarkkisessa mallissa jokaiselta sivulta on linkki alemmalla tasolla sijaitsevalle sivulle, mutta ylöspäin ei ole mahdollista palata. Verkkomallisella sivustolla sivuilta toiselle voidaan liikkua edestakaisin /14, s.17/. GateKeeper-ohjelman suunnittelussa päädyttiin verkkomalliseen ratkaisuun. Sivumäärän pysyessä suhteellisen pienenä,

verkkomallisen ratkaisun uskotaan lisäävän käyttömukavuutta. Samalla sivulla pystytään toteuttamaan useita eri toimintoja, eli sivuja tarvitsee ladata melko harvoin.



Kuva 3. GateKeeper-ohjelman käyttöliittymä muodostuu toisiinsa linkitetyistä web-sivuista

Käyttäjälle näkyvät web-sivut ovat moduuleita, jotka sisältävät runsaasti toiminnallisuutta (liite 1). Sivuille sijoitetut HTML-elementit tekevät niistä dynaamisia, eli suorittavat pyydettyjä toimintoja. Ohjelmalle välitetään syötteitä HTML-lomakkeiden avulla. Lomakkeet sisältävät elementtejä kuten tekstiruutuja, valintaruutuja, valintanappeja, luetteloruutuja ja piilokenttiä. Lomakkeelle määritellään, mikä sivu web-sovelluksessa käsittelee sille syötetyt tiedot. Kun tietokanta kytketään osaksi verkkopalvelua, voidaan graafinen näkymä ja tietovarasto erottaa toisistaan. Käyttöliittymästä annetaan syötteitä ja komentoja sovelluslogiikalle, joka noutaa tai tallentaa tiedon tietovarastoon.

4.4 Sivujen suojaus ja käyttöoikeudet

Annettujen rajoitusten mukaan GateKeeper-ohjelmistoon kuuluvien sivujen käyttö on sallittu vain niille käyttäjille, joilla on hallussaan voimassaoleva käyttäjätunnus, salasana ja käyttäjärooli. Toisin sanoen, ohjelmiston käyttäminen edellyttää kirjautumista. Ilman voimassaolevia tunnistetietoja sivuille ei ole pääsyä. Ohjelmisto on suunniteltu niin, että käyttäjä palautetaan aina takaisin kirjautumissivulle, mikäli hän yrittää ladata ohjelmistoon kuuluvia sivuja ilman kirjautumista. Päästäkseen ohjelmistoon kuuluville muille sivuille, joutuu käyttäjä ennen sitä antamaan ohjelman kirjautumissivulla voimassaolevan käyttäjätunnuksen, salasanan sekä hänelle määritetyn käyttäjäroolin.

Ohjelman suunnitteluvaiheessa sivujen suojaus päätettiin toteuttaa valmiin SimpleAuth-käyttäjäntunnistusmekanismin avulla /15/. Ohjelma tarkistaa annetut tiedot GateKeeper-tietokannasta ja ohjaa käyttäjän hänen roolinsa mukaiselle aloitussivulle. Ohjelmaan on määritetty eritasoisia käyttöoikeuksia omaavia käyttäjäryhmiä. User-ryhmään kuuluvilla käyttäjillä on ohjelmassa oikeus vaihtaa vain oma salasanansa. Administrator-ryhmään kuuluvilla käyttäjillä on sen sijaan useita, kaikkia käyttäjiä koskevia oikeuksia. Käyttäjäryhmät on määritelty MySQL-tietokannassa, josta ohjelma tarkistaa käyttäjän roolin. Mikäli kirjautumissivulla annetut tiedot ovat oikein, ohjelma luo niiden perusteella istunnon *(session).* Jokaisen sivun lataamisen yhteydessä tarkistetaan, onko istunto voimassa. Mikäli tarkistusvaiheessa todetaan, ettei näin ole, palautetaan käyttäjä takaisin kirjautumissivulle.

Istunto on tapa tallentaa määrättyjä tietoja web-sivun lataamisen yhteydessä. /16./ Näitä tietoja voidaan käyttää hyväksi, kun sivu ladataan uudelleen. Tietoja tallentamalla on mahdollista toteuttaa kehittyneempiä ja käyttäjäystävällisempiä web-sivuja. Istunnon luomisen yhteydessä käyttäjälle annetaan yksilöllinen istuntotunnus *(session id). Se* tallennetaan käyttäjän koneella olevaan tiedostoon, eli evästeeseen *(cookie),* tai välitetään palvelimelle URL-osoitteen mukana. Kun web-sivun ohjelmakoodissa on istunto-moduuli otettu käyttöön, tarkistetaan aina sivun lataamisen yhteydessä, onko istuntotunnus välitetty palvelupyynnön mukana. Tarkistus tehdään suoraan *session_start()*-funktiolla, tai epäsuorasti *session_register()*-funktion avulla. Mikäli istunto on voimassa, palautetaan aikaisemmalla kerralla tallennetut tiedot. Tässä tapauksessa tämä tarkoittaa, että palvelin tunnistaa kirjautuneen käyttäjän.

4.5 Tietokanta

Moodle, Mambo ja Subversion ovat ohjelmia, joita käytetään Karttakeppi.com oppimisympäristön kehitystyössä. Näistä ohjelmista sekä Moodle että Mambo tallentavat ohjelmien käyttäjätiedot MySQL-tietokantaan, jonka asennusohjelma on luonut automaattisesti palvelimelle.

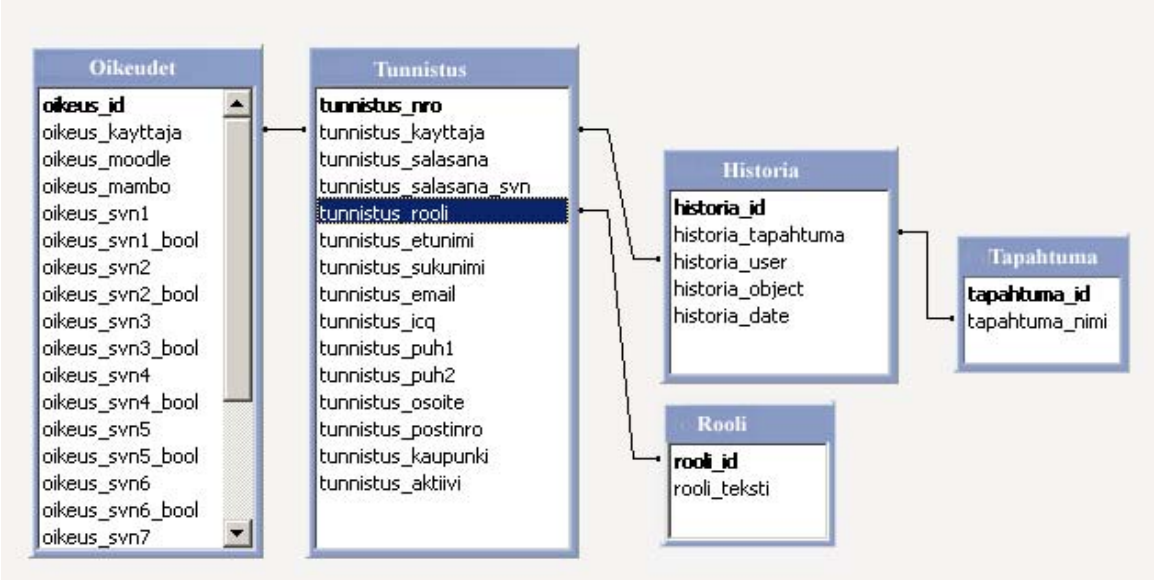
Koska Karttakeppi.comin palvelimella on jo toimiva MySQL -tiedonhallintajärjestelmä, on perusteltua käyttää sitä myös GateKeeper-ohjelmassa tallennettavien tietojen tietovarastona. Periaatteessa mikä tahansa muukin tiedonhallintajärjestelmä voisi toimia GateKeeper-ohjelman tietovarastona.

MySQL on ei-kaupallisessa käytössä ilmainen ja suorituskykyinen relaatiotietokanta. Relaatiotietokanta perustuu relaatiomalliin, jossa kaikki tieto tallennetaan tauluihin. Samassa tietokannassa voi olla useita tauluja. Taulussa yhteen kohteeseen liittyvät tiedot muodostavat rivin eli tietueen. Kohde on tunnistettavissa annetun perusavaimen perusteella, sillä kahdella tai useammalla tietueella ei voi olla samaa perusavaimen arvoa. MySQL käyttää tietojen hakemiseen, käsittelyyn ja lisäämiseen SQL-kieltä. GateKeeper-projektissa käytetty PHP-ohjelmointikieli sisältää paljon valmiita funktioita, joiden avulla MySQL-tietokantaa voidaan käsitellä. PHP-koodiin on myös helppo liittää SQL-kyselyitä, joiden avulla tietokannassa olevia tietoja voidaan lukea sekä muuttaa.

GateKeeper-ohjelmisto vaatii toimiakseen luku- ja kirjoitusoikeudet ohjelmaa varten perustettuun MySQL-tietokantaan. Ohjelmassa on toimintoja, joiden avulla käyttäjä voi lukea, muuttaa ja lisätä tietoja tietokantaan. Tapahtumista kerätään myös historiatietoja, jotka tallennetaan samaan tietokantaan. Historiatietojen avulla voidaan seurata tietoihin tehtyjä muutoksia.

Oikein suunnitellussa tietokannassa ei saa olla päällekkäisyyttä *(redundassia)* tiedoissa. Päällekkäisyys aiheuttaa ylimääräistä tietojen tallentamis- ja päivittämistyötä. Tällöin myös virheiden määrä kasvaa. Tietojen muutos ja lisäys pitäisi tehdä vain yhteen paikkaan, eikä siitä saisi aiheutua ristiriitoja tai tarpeellisten tietojen tuhoutumista. /17./ GateKeeper-tietokanta on pyritty suunnittelemaan näiden periaatteiden mukaisesti. Siinä on otettu huomioon myös ohjelmiston laajennusmahdollisuus. Kuvassa 4 on esitetty GateKeeper-tietokannan taulut sekä niiden väliset yhteydet.

Tunnistus-taulu sisältää käyttäjien perustiedot. Tunnistus_nro-kenttä on taulun perusavain. Tunnistus_kayttaja-kenttään ei haluta samoja arvoja, eli jokaisella käyttäjällä täytyy olla yksilöllinen tunnus. Sen vuoksi tunnistus_kayttaja-kentälle on asetettu ehto *unique*. Käyttäjien salasanat eivät näy salasana-kentässä selväkielisinä, vaan ne salataan tallennusvaiheessa md5-funktiolla. Tämän vuoksi kentän tyypiksi on määritetty MySQL-sivuilla esiintyvän ohjeen mukaisesti *tinyblob*. Muut tunnistus-taulun kentät vastaavat useimmiten niitä kenttiä, jotka esiintyvät myös Moodle- ja Mambo -tietokannoissa. Tietojen kopiointi muiden ohjelmien tietokantoihin helpottuu, jos tietokantojen tyypit ovat mahdollisimman samankaltaisia.



Kuva 4. GateKeeper-tietokannan taulut ja niiden väliset yhteydet

5 TIETOTURVAUHKIA JA -RATKAISUJA

5.1 Web-palveluihin liittyviä riskejä

Internetin välityksellä tieto on käytettävissä missä ja milloin tahansa. Tämän merkittävän edun haittapuolena on, että tieto saattaa joutua myös niille, joille sitä ei ole tarkoitettu. Web-palveluiden yhteydessä halutaan yleensä määrätyille käyttäjille tarjota valvottu ja rajoitettu oikeus käyttää verkkoa ja sen palveluita. Mikäli käyttäjien oikeuksia ei rajata huolellisesti, web-palvelimella alkaa todennäköisesti pian esiintyä ei-toivottua toimintaa. Seurauksena saattaa olla sekä tietoturvan vaarantuminen että häiriöitä palvelimen toiminnassa.

Tietoturvan ylläpito on sekä palvelimen ylläpitäjän, että ohjelmistosuunnittelijoiden yhteinen huolenaihe /18/. Palvelimen ohjelmistot on pidettävä jatkuvasti ajantasalla ja ohjelmista löydetyt tietoturva-aukot päivitettävä mahdollisimman nopeasti. On tärkeää ymmärtää, että vasta perustettu web-palvelin on alttiina hyökkäyksille heti sen ensi hetkistä alkaen. Internetissä on runsaasti tahoja, joiden tarkoituksena on löytää web-palvelimesta paikalliseen verkkoon johtavia aukkoja. Myös itse palvelinta yritetään käyttää epätarkoituksenmukaisesti, jopa rikolliseen toimintaan. Verkon ylläpitäjien tavoite on pitää tunkeilijat oman verkon ulkopuolella. Sen vuoksi palvelimen käyttöoikeuksien määrittely ja käyttäjien tunnistus ovat erittäin tärkeitä, tietoturvaa lisääviä tekijöitä.

5.2 Web-palveluiden suojausmenetelmät

Web-palvelimen ja lähiverkon turvaamiseksi tunkeilijoita vastaan on olemassa useita keinoja. Yksi yleisimmin käytetyistä suojausmenetelmistä on IP-osoitteen perusteella tapahtuva käytönvalvonta. Tämä tarkoittaa, että tulevat palvelupyynnöt voidaan hyväksyä tai hylätä palvelupyynnön lähettäjän IP-osoitteen perusteella. IP-osoitteiden tarkistus ja suodatus voidaan tehdä palvelinohjelmistoilla, kuten Apache tai IIS *(Microsoft Internet Information Server)*. IP-osoitteiden suodatus *(blocking)* on yksinkertainen ja helppo toimenpide, joka ei edellytä toimenpiteitä palvelun käyttäjältä. Koska suodatus määritellään tietylle sivustolle, IP-suodatuksen haittapuolena on pääsyn esto web-palvelun lisäksi myös kaikille muille kyseisen sivuston sivuille. Palvelimen ilmoitusta palvelu-pyynnön hylkäämisestä ei myöskään voida pitää erityisen käyttäjäystävällisenä. Pääsy sivuston muille sivuille voidaan tosin sallia IP-suodatuksen yhteydessä luomalla

palvelimelle virtuaalihakemistoja. Sitä vastoin IP-osoitteiden väärennystä *(spoofing)* on vaikeampi estää, mikä on omiaan heikentämään IP-suodatuksella tapahtuvan suojauksen luotettavuutta. /19./

IP-osoitteiden suodatusta joustavampi tapa suojata web-palveluja on käyttäjän todentaminen tapauskohtaisesti. Tämän tyyppinen suojauskäytäntö mahdollistaa palvelun käytön mistä tahansa ja käyttäjä saa ongelmatilanteessa täsmennetyn virheilmoituksen. Käyttäjän todentamiseen perustuva suojauskäytäntö on yleensä työläämpi toteuttaa, mutta sen avulla voidaan myös kerätä erilaisia palvelun käyttöön liittyviä tietoja. Tähän suojauskäytäntöön liittyvät tietoturvariskit aiheutuvat yleensä tarpeesta tallentaa arkaluonteisia käyttäjätietoja, kuten salasanoja ja käyttäjätunnuksia. /19./

Web-palvelujen suojaukseen on olemassa myös kolmas tapa, joka perustuu digitaalisiin allekirjoituksiin *(digital certificates).* Sertifikaatti sisältää tietoja henkilöstä, jolle sertifikaatti on myönnetty, sekä tietoja sertifikaatin myöntäjästä. Tällä suojauskäytännöllä voidaan toteuttaa joustava ja tapauskohtainen tunnistusmekanismi, jonka avulla käyttäjätietojen kerääminen on mahdollista. Tähän suojausmenetelmään liittyvät ongelmat aiheutuvat asiakaskoneelle tehtävästä asennuksesta. Asennus ei välttämättä ole yksinkertainen. Silti sertifikaatti on asennettava jokaiselle koneelle jolla web-palvelua halutaan käyttää. /19./

5.3 Dynaamisten HTML-lomakkeiden väärinkäyttö

PHP-kieli antaa käyttäjille mahdollisuuden suorittaa komentoja palvelimella, sekä käyttää palvelimella olevia tietoja ja tietoliikenneyhteyksiä. Tietojen ja komentojen syöttö palvelimelle tapahtuu yleensä HTML-lomakkeiden avulla. PHP-kieli tarjoaa joustavia funktioita tiedostojen käsittelyyn *(flexible file handling functions)*. Näiden avulla voidaan käsitellä sekä paikallisia että URL-osoitteella varustettuja tiedostoja. PHP-sovelluksiin erikoistuneen yrityksen Zendin mukaan monet PHP:n tietoturvariskit johtuvat dynaamisten tiedostojen ja tiedostopolkujen väärästä käsittelystä. Esimerkkinä Zend mainitsee osoitteen, joka saattaisi näkyä selaimen osoiterivillä muodossa:

http://example.com/page.php?i=aboutus.html

Tässä tapauksessa hyökkääjän olisi helppo muuttaa osoitetta esimerkiksi niin, että *i:n* tilalle tulisikin*: i=../../../etc/passwd?.* Tämän kaltaisia virheitä olisi hyvä oppia välttämään.

Monissa lähteissä neuvotaan sijoittamaan dynaamisesti käsiteltävät tiedostot paikkaan, jossa niitä ei normaalisti nähdä web-selaimen avulla. Syötteissä annettujen sivujen käsittely voidaan estää määrittelemällä käsiteltävät sivut jo ohjelmakoodissa. PHP-koodin include-tiedostojen olisi myös oltava .php-päätteisiä niiden lähdekoodin tarkastelun estämiseksi.

CGI-ohjelmat voivat suorittaa palvelimella periaatteessa mitä tahansa komentoja web-käyttäjän oikeuksilla. Sen vuoksi palvelimelle annetut komennot tulee tarkastaa ja niille tulee asettaa rajoituksia. Monet rajoituksista asetetaan jo palvelimen määrittely-tiedostoissa, mutta yhtä tärkeää on tarkastaa käyttäjän syötteet myös ohjelmakoodissa. Toisin sanoen, on aina syytä epäillä, että käyttäjä toimii jollakin odottamattomalla tavalla ja antaa syötteissä mitä tahansa mielivaltaisia komentoja. Lähtökohtana pitäisi aina olla syötteiden tarkistaminen ja niiden muuntaminen harmittomaan muotoon.

GateKeeper-ohjelmassa käytetään runsaasti lomakkeita, joihin tiedetään liittyvän myös runsaasti riskejä. Liian luottavainen ja välinpitämätön ohjelmointitapa voi jättää vakavia turvallisuusaukkoja, joita epätoivotut tahot pyrkivät hyödyntämään. Jos hyökkääjällä olisi tilaisuus lähettää lomakkeella mitä tahansa merkkijonoja, kuten SQL-kyselyitä ja HTML-komentoja, olisi hänen helppo sekoittaa koko tietokanta tietokantakäskyjen *(SQL Injection attacks)* avulla. Kun käyttäjän tunnistuksessa käytetään apuna tietokantaa, voidaan tieto-kantakäskyillä jopa murtautua salasanalla suojatulle sivustolle. Microsoftin tietoturvasivulla aiheesta on annettu lyhyt esimerkki. Esimerkissä käyttäjän odotetaan antavan käyttäjä-tunnus ja salasana eri kenttiin. Oikein täytettynä lomakkeelta lähetetty kysely näyttäisi tältä:

*SELECT * FROM Users WHERE UserName='Paul' AND Password='password'*

Oikean tunnuksen sijasta hyökkääjä antaakin syötteeksi *' Or 1=1* --, jolloin kysely saa muodon:

*SELECT * FROM Users WHERE UserName=' ' Or 1=1 --' AND Password=' '*

Kahden tavuviivan tarkoittaessa kommentin alkua, kysely tulkitaan edelleen muotoon:

SELECT FROM Users WHERE UserName=' ' Or 1=1*

Koska lause *'1=1'* on aina tosi, kysely tuottaa tuloksen aina kun *'Users'* -taulussa on yksikin rivi. Pahimmillaan hyökkääjä pystyisi myös helposti poistamaan kaikki *Users*-taulussa olevat tiedot kirjoittamalla *Username*-kenttään*: Paul'; drop table Users--.* Onneksi tämän kaltaisiin hyökkäyksiin on kuitenkin kehitetty myös tehokkaita puolustuskeinoja.

Annettujen syötteiden merkkejä voidaan tarkistaa kahdella tavalla. Syötteistä voidaan etsiä kiellettyjä merkkejä, mutta toisaalta voidaan tarkistaa että syötteet sisältävät vain sallittuja merkkejä. Sallittujen merkkien määrittelyyn käytetään yleisesti säännöllisiä lausekkeita *(regular expressions).* PHP-kieli tarjoaa keinoja, joilla pystytään automaattisesti estämään tuhoa aiheuttavia syötteitä. /20./ Virheellisten syötteiden hylkäyksen yhteydessä puhutaan yleensä escaping-menetelmistä. Asettamalla palvelimelle *magic_quotes_gpc*-asetus käyttöön, voidaan erilaisilla lainausmerkeillä sotketut syötteet välittömästi hylätä. Jos *macic quotes* on pois käytöstä, joudutaan itse huolehtimaan kenoviivojen lisäämisestä merkkijonoissa olevien lainausmerkkien eteen *addslashes($merkkijono)*-funktion avulla ennen tietojen tallentamista tietokantaan.

Tiedonhallintajärjestelmillä on myös omia suojausmenetelmiä. MySQL osaa koodata automaattisesti kaiken lomakkeilta annetun tiedon, mikä estää tiedon korruptoitumista tietokannassa. MySQL ei hyväksy myöskään useita tietokantakomentoja samassa syötteessä, kuten esimerkiksi SQLite and PostgreSQL. MySQL-tietokannan yhteydessä annetun merkkijonon tarkistukseen on myös mahdollista käyttää *MySQL_real_escape_string()*-funktiota. Sitä käytettäessä on kuitenkin tarkistettava, onko *magic_quotes* -asetus päällä. Tässä tapauksessa kaksinkertainen tarkastus korruptoisi syötteen.

Olennaista kuitenkin on, että web-palvelulle on annettu vain tarvittavat oikeudet tietokannan käyttöön. Toinen, merkittävä tietoturva-aukkoja ehkäisevä keino on asettaa *php.ini* – tiedostossa *register_globals* asetus off-tilaan. Tällä asetuksella estetään GET- ja POST-metodien sekä *Cookie*, *Server*, *Environment* ja *Session*-muuttujien toimiminen globaalisti ohjelmassa. Silloin voidaan lisäksi varmistua, mistä käytetyt muuttujat ovat todella peräisin. Monesti hyökkääjä pyrkii selvittämään lomakkeenkäsittelijän sijainnin ja ottamaan sen omaan käyttöönsä. Lomakkeenkäsittelijän joutuminen hyökkääjän välityspalvelimeksi estetään varmistamalla lomakkeelle tulevan tiedon alkuperä. Toisin sanoen, olisi selvitettävä miltä palvelimelta ja mistä domainista lomakkeenkäsittelijää yritetään käyttää. Turvallisuutta saadaan lisättyä hyväksymällä vain omalta palvelimelta lähtöisin oleva tieto. *HTTP_REFERER*-tarkistus estää lomakkeenkäsittelijää prosessoimasta ja lähettämästä eteenpäin sellaista tietoa, joka ei ole peräisin omalta palvelimelta /20/.

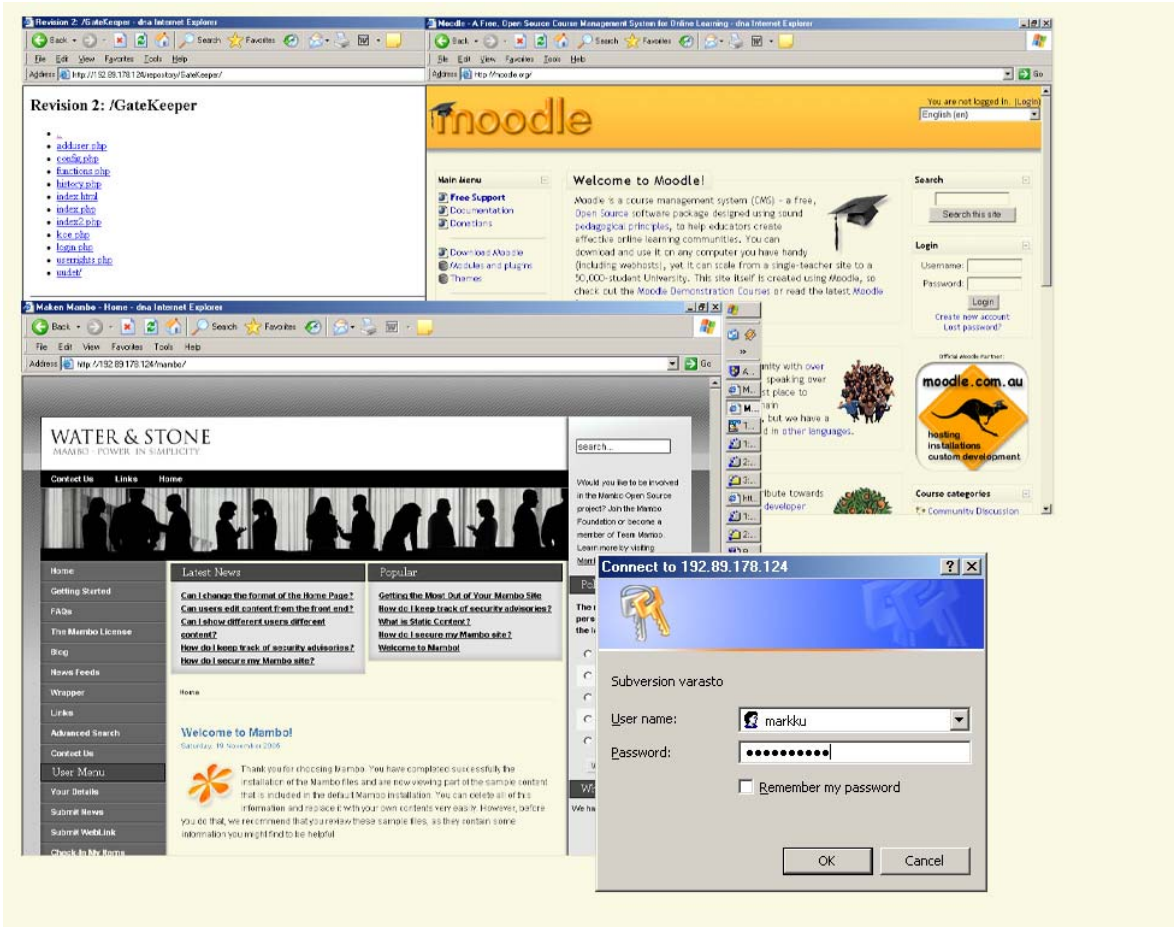
5.4 Tietoturvallinen ohjelmistokehitys

GateKeeper-ohjelman kehityksen yhteydessä tietoturva on ollut yksi tärkeimmistä lähtökohdista. Tietoturvallisuudesta huolehtiminen ja ohjelman käyttömukavuuden säilyttäminen ovat tasapainoilua molempien tavoitteiden yhteensovittamiseksi, mikä ei ole kuitenkaan mahdoton tehtävä. Tässä luvussa mainittujen tietoturvanäkökohtien huomioimiseksi ohjelman testaussuunnitelmaan lisättiin vaihe tietoturvallisuuden tarkistamista varten.

Tietoturvallisuuden kannalta valittu ohjelmointikieli ei välttämättä ole ratkaisevin tekijä. PHP-kieli kuten mikään muukaan ohjelmointikieli ei ole täydellisen haavoittumaton /20/. Tiettyjä eroja ohjelmointikielien välillä toki löytyy. Esimerkiksi C ja C++ kielet mahdollistavat muistipuskurin ylivuodon, kun taas Javassa muistinhallinta on osa järjestelmää. Olennaista on, että ohjelmointikielen taustalla on taho, joka pyrkii kehittämään kieltä ja pystyy paikkaamaan nopeasti löydetyt tietoturva-aukot. Lopullinen vastuu jää kuitenkin ohjelmiston suunnittelijalle ja ohjelmiston ylläpitäjälle. Koska täydellistä tietoturvaa ei voida saavuttaa, on tarpeellista kartoittaa mahdolliset tietoturva-uhat. Jokaiselle tunnistetulle uhalle on tehtävä myös jokin vastatoimenpide uhkaa pienentämään. Tietoturvallisessa ohjelmistokehityksessä ohjelma suunnitellaan siten että se toimii oikein myös pahantahtoisen hyökkäyksen sattuessa. Tietoturvallisuusnäkökohdat on otettava huomioon heti ohjelmiston suunnittelun alkuvaiheissa, sillä jälkikäteen varmuuden saaminen on enää vaikeaa. Suunnittelussa tapahtuneita virheitä voidaan löytää usein testaamalla ja testaus edellyttää hyvää dokumentointia. Käyttämällä parhaita kehityskäytäntöjä, pystytään merkittävästi parantamaan ohjelmistojen turvallisuutta. /21./

6 INTEGROITAVAT OHJELMAT

GateKeeper-ohjelmistoprojektin ensimmäisessä vaiheessa tavoitteeksi asetettiin kolmen eri ohjelman integroiminen GateKeeper-ohjelmaan siten, että näiden ohjelmien käyttäjätietojen hallinta olisi mahdollista keskitetysti. Valitut ohjelmat ovat Mambo, Moodle ja Subversion (kuva 5).



Kuva 5. GateKeeper-ohjelmaan ingroidut ohjelmat

6.1 Moodle, oppimisympäristö

Moodle on ilmainen, avoimen lähdekoodin ohjelmisto, jonka avulla valmiita kursseja voidaan julkaista Internetissä /22/. Ohjelmisto on suunniteltu opetusta, tiedottamista, yhteydenpitoa sekä sähköisen materiaalin jakamista varten. Moodlea käytetään esimerkiksi oppilaitoksissa, yrityksissä, yhteisöissä ja seuratoiminnassa.

Moodle on PHP-kielellä toteutettu, yleisimmällä selaimella toimiva ohjelma. Moodlen tietovarastona voidaan käyttää MySQL-, PostgreSQL-, Oracle-, Access-, Interbase-, tai ODBC-rajapinnan mukaisia tiedonhallintajärjestelmiä. Ohjelma luo asennuksen yhteydessä automaattisesti tietokannan valittuun tiedonhallintajärjestelmään.

Moodlen asennus tapahtuu yksinkertaisesti kopioimalla Moodle-tiedostot web-palvelimelle paikkaan, josta ne voidaan esittää yleisölle. Paikka voi olla esimerkiksi muotoa: *http://palvelimenOsoite/moodle.* Tähän osoitteeseen ensimmäisellä kerralla siirryttäessä käynnistyy ohjattu toiminto, joka suorittaa Moodlen asennuksen. Asennusohjelma tarkistaa web-palvelimen määritykset, luo tietokannan sekä *config.php* -tiedoston, joka sisältää ohjelman tärkeimmät asetukset. Tässä vaiheessa asennusohjelma kysyy tietokannan käyttämiseen oikeuttavan tunnuksen ja salasanan.

Moodlen asennuksen jälkeen järjestelmän valvoja voi tehdä vielä monia ohjelman toimintaan liittyviä asetuksia. GateKeeper-ohjelman kannalta asetuksista olennaisin on valittu käyttäjäntunnistusmetodi. Valittavissa on useita metodeja. Käyttäjien tunnistukseen voidaan käyttää Moodlen omia käyttäjätilejä tai ulkoista tietokantaa. Mikäli tunnistukseen valitaan ulkoinen tietokanta, täytyy Moodle-ohjelmalle kertoa tietokannan käyttöön oikeuttavat tunnukset sekä käyttäjätietoja sisältävien taulujen nimet.

Ulkoisen tietokannan käyttö Moodlen käyttäjien tunnistukseen olisi yksinkertaista, mutta siihen liittyy ongelmia. GateKeeper-projektissa pyritään käyttäjätietoja hallinnoimaan keskitetysti ja oikeuksia myöntämään vain tarvittaessa. Sen vuoksi Moodle ei voi tarkistaa käyttäjätietoja suoraan GateKeeper-tietokannasta. Toisin sanoen, oikeus GateKeeper-ohjelman käyttöön ei saa antaa oikeutta automaattisesti Moodle-ohjelman käyttöön. Tämän vuoksi Moodle-ohjelman käyttäjien tunnistuksessa päädyttiin käyttämään ohjelman omia käyttäjätilejä. GateKeeper-ohjelma toimii niin, että uuden käyttäjätunnuksen luomisen yhteydessä sama tunnus lisätään myös Moodle-tietokantaan. Tässä vaiheessa Moodle-tietokantaan talletetaan kuitenkin oikean salasanan sijasta arvottu salasana, jota käytännössä ei voi arvata. Luodulla käyttäjätunnuksella ei siis voida vielä käyttää ohjelmaa. Oikea salasana tallennetaan Moodle-tietokantaan vasta kun Moodlen käyttö-oikeus myönnetään erikseen GateKeeper-ohjelmassa. Mikäli käyttäjältä evätään Moodlen käyttöoikeus, tallennetaan Moodle-tietokantaan jälleen arvottu salasana, jolloin mainitulla käyttäjätunnuksella ei ole enää mahdollista käyttää ohjelmaa.

6.2 Mambo, dynaaminen julkaisujärjestelmä

Mambo on ilmainen, GLP-lisenssin alainen ohjelma /23/. Mambon ideana on tarjota ratkaisu web-pohjaisten palvelujen ylläpitoon ja päivitykseen ilman ohjelmointiosaamista. Sitä voidaan käyttää paitsi yksinkertaisten web-sivujen, myös laajojen yritysratkaisujen hallintaan. Mamboon on saatavissa valmiita lisäkomponentteja, kuten kuvagalleria, keskustelufoorumi tai vieraskirja. Mambo on ohjelmisto, joka koostuu dynaamisista, PHP-kielellä toteutetuista web-sivuista.

Moodlen lailla Mambo asennetaan myös kopioimalla sen tiedostot ensin web-palvelimelle paikkaan, jossa ne voidaan esittää yleisölle. Asennusohjelma käynnistyy, kun Mambo-sivuja ladataan ensimmäistä kertaa. Asennusohjelma tarkistaa palvelimen asetukset, luo tietokannan, sekä tallettaa asetukset *configuration.php*-tiedostoon. Sivusto on käytettävissä heti asennusohjelman suorituksen ja *installation*-hakemiston poiston jälkeen.

Mambon käyttäminen vaatii myös käyttäjätunnuksen lisäksi salasanan, jotka löytyvät Mambon omasta tietokannasta. Mambo käyttää aina MySQL -tyyppistä tietokantaa. Tämä mahdollistaa tietojen kopioimisen GateKeeper-tietokannasta Mambon tietokantaan suhteellisen helposti. GateKeeper-ohjelma tallentaa käyttäjätunnuksen Mambon tietokantaan heti käyttäjätunnuksen luomisen yhteydessä. Tässä vaiheessa Mambon tietokantaan tallennetaan jälleen arvottu salasana. Uusi käyttäjä ei siis pysty vielä käyttämään Mamboa ennen kuin hänelle myönnetään käyttöoikeus erikseen GateKeeper-ohjelmalla. Arvottu salasana korvataan oikealla salasanalla, kun Mambon käyttöoikeus myönnetään käyttäjälle. Mikäli käyttäjältä evätään Mambon käyttöoikeus, tietokantaan talletetaan jälleen arvottu salasana.

6.3 Subversion, versionhallintaohjelma

Subversion on palvelimelle asennettu ohjelmisto, joka on tarkoitettu tiedostojen versionhallintaan /24/. Ohjelman avulla tiedostoihin tehdyt muutokset voidaan jäljittää, sekä tarvittaessa palauttaa voimaan tiedoston vanhempi versio. Käyttäjät päivittävät yhteistä hakemistoa Subversion-asiakasohjelman avulla. Asiakasohjelma ilmoittaa, mikäli yhteisessä hakemistossa oleviin tiedostoihin on tullut muutoksia. Siten käyttäjä voi varmistua, että hänellä on aina käytössään tiedoston viimeisin versio.

Subversion toimii siirrettävällä kerroksella, joka käyttää APR-rajapintaa *(Apache Portable Runtime library).* Versiohistoriaan päästään käsiksi selaimella tai yhteensopivalla asiakasohjelmalla. Versiohistorian ydin on varastokirjasto *(repository)*, eli tiedon keskitetty sijoituspaikka. Varastokirjastossa tiedostot ja hakemistot talletetaan hierarkkisessa järjestyksessä hakemistopuuhun. Tiedostonhallintajärjestelmäksi voidaan valita joko Berkeley DB, tai käyttöjärjestelmän alkuperäistä tiedostojärjestelmää noudattava FSFS.

GateKeeper-ohjelman toteutuksessa Subversion integroitiin osaksi Apache-palvelinohjelmistoa. Vaikka Subversion pystyy toimimaan yhdessä Apachen kanssa, se ei kuitenkaan ole siitä täysin riippuvainen. APR-rajapinta käsittää kattavan kirjaston, joka on kaikkien sovellusten käytettävissä. Jotta se toimisi Apachen kanssa, joudutaan määrittelytiedostoon *(httpd2.conf)* lisäämään rivi *LoadModule* -direktiiviä varten. Jotta Apache pystyisi ohjaamaan svn-alkuisten osoitteiden käsittelyn Subversion-ohjelmalle, tiedostoon lisätään myös jälkimmäinen <*Location*> -määritelmä:

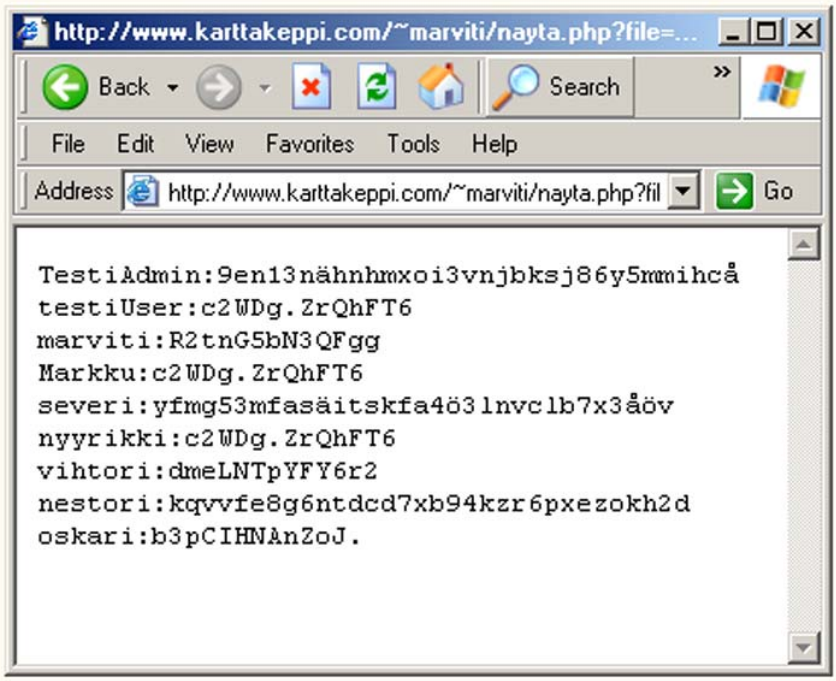
```
LoadModule dav_svn_module          modules/mod_dav_svn.so

<Location /repository>
DAV svn
SVNPath /home/repos
AuthType Basic
AuthName "Subversion varasto"
AuthUserFile /etc/svn-auth-file
Require valid-user
</Location>
```

Määritelmä osoittaa Subversion-varastokirjaston sijainnin ja edellyttää käyttäjän tunnistamista HTTP-autentikaation avulla *(HTTP Basic authentication mechanism).* HTTP-autentikaatio toimii siten, että käyttäjän siirtyessä suojatulle sivulle palvelinohjelmisto pyytää käyttäjän tunnusta ja salasanaa. Näitä tietoja verrataan <*Location*>-määritelmässä osoitetun salasanatiedoston vastaaviin tietoihin. Normaalisti käyttäjä luo ja päivittää salasanatiedoston Apachen htpasswd-työkalun avulla. Tarkasteltaessa salasanatiedostoa havaittiin käyttäjätunnuksen ja salatun salasanan olevan siellä muodossa (kuva 6):

```
tunnus:salasana_salatussa_muodossa
```

Projektissa selvitettiin miten tunnus ja salasana olisi mahdollista erottaa toisistaan ja miten tiedostoa voidaan päivittää GateKeeper-ohjelman avulla. Tarkastelun jälkeen todettiin, että syötetty salasana voidaan salata salasanatiedostoa varten PHP-funktiolla:

$salasana_salatussa_muodossa =crypt($salasana, base64_encode($salasana));



Kuva 6. Subversion-ohjelma tarkastaa käyttäjätunnukset ja salasanat kuvan kaltaisesta salasanatiedostosta.

Luvussa 7 kerrotaan miten GateKeeper-ohjelma lukee salasanatiedoston ja pystyy lisäämään siihen uuden käyttäjätunnuksen ja salasanan. Lisäksi ohjelman avulla voidaan muuttaa tietyn käyttäjän salasanaa ja tallentamaan se tiedostoon vanhan salasanan tilalle.

7 TOTEUTUS

7.1 SimpleAuth-suojausmekanismi

GateKeeper-ohjelmiston sivujen suojaus perustuu Jez Hancockin SimpleAuth-tunnistusmekanismiin, jonka toiminnan yleispiirteet on kerrottu luvussa 4.4.

Kirjautuminen GateKeeper-sivustolle tapahtuu aina index.php-sivun kautta. Mikäli sivustolle yritetään ilman kirjautumista, palauttaa selain automaattisesti näkymän takaisin index.php-sivulle. Kirjautumissivun nimeksi on valittu index.php, koska näin estetään hakemistorakenteen tarkastelu selaimella. Palvelinohjelmiston asetuksista riippuen palvelin lähettää automaattisesti index-nimisen sivun, mikäli siltä pyydetään hakemiston osoitetta ja index-niminen sivu löytyy hakemistosta.

Kaikille GateKeeper-ohjelman näkyville sivuille on lisätty SimpleAuth-mekanismin mukainen tarkistusosio, jossa käyttäjän rooli ja kirjautumisen tila tarkistetaan (liite 2):

```
if(($_SESSION["rooli"]!='1'))
                {
                header("Location: index.php?");
                }
tarkistaKirjautuminen("yes");
```

Tässä esimerkissä käyttäjä ohjataan kirjautumissivulle, mikäli muuttujaa *”$_SESSION["rooli"]”* ei ole alustettu, tai käyttäjällä ei ole pääkäyttäjän roolia, jota vastaa arvo ”1”. Funktio tarkistaKirjautuminen(”yes”) tekee saman, mikäli muuttujaa *$_SESSION[”kirjautunut”]* ei ole alustettu. Tämä tarkastetaan tarkistaKirjautuminen()-funktion case-haarassa liitteen 2 mukaisesti:

```
case "yes":
if(!isset($_SESSION["kirjautunut"])){
header("Location: index.php");
exit;
}
break;
```

$_SESSION*-muuttujat alustetaan *luoTodennettuIstunto($tunnus,$salasana,$rooli) -funktion avulla, jota kutsutaan kirjautumissivulla. Sivulla on lomake, johon käyttäjä

syöttää käyttäjätunnuksen, salasanan sekä käyttäjäroolin. Lomakkeella annetut arvot välitetään funktiolle *$_SERVER["PHP_SELF"]* -superglobaalimuuttujan avulla.

```
function luoTodennettuIstunto($tunnus, $salasana, $rooli) {
$_SESSION["tunnus"]=$tunnus;
$_SESSION["salasana"]=$salasana;
$_SESSION["kirjautunut"]=true;
$_SESSION["rooli"]=$rooli;
}
```

Ennen kuin ohjelma välittää kirjautumissivun lomakkeella annetut tiedot eteenpäin *luoTodennettuIstunto()*-funktiolle, syötetty tunnus ja salasana tarkistetaan *kentanTarkistus()*-funktiolla. Tässä vaiheessa käyttäjälle ilmoitetaan, mikäli annetut tiedot eivät pituudeltaan tai merkistöltään vastaa asetettuja vaatimuksia. Tiedot välitetään eteenpäin vasta kun *tarkistaTunnukset($tunnus, $salasana, $rooli)* -funktio on verrannut annettuja tietoja GateKeeper-tietokannassa oleviin tietoihin ja löytänyt sieltä täsmälleen yhden vastaavan rivin:

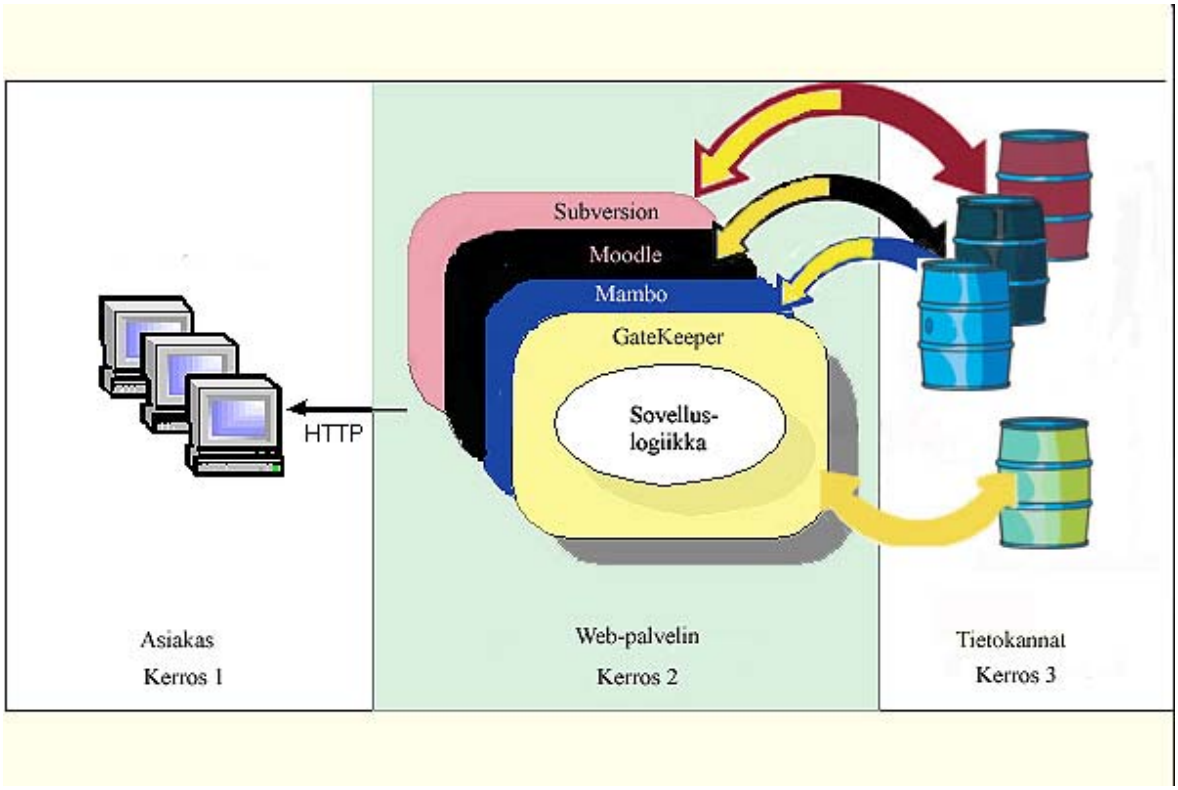
```
$kysely="SELECT tunnus, salasana rooli from tunnistus WHERE....;
$vastaus=MySQL_query($kysely, $gatekeeper_yhteys);
if(MySQL_num_rows($vastaus)==1)
                {
                $rivi=MySQL_fetch_array($vastaus);
                return $rivi;
                }
```

SimpleAuth-mekanismiin kuuluu myös config.php-moduuli, jossa määritellään suojausmekanismiin liittyviä asetuksia. Tiedostossa rekisteröidään istunnon globaalit muuttujat *session_register()*-funktion avulla, sekä määritellään tarvittavat parametrit tietokantayhteyden avaamista varten.

Web-palvelimen php.ini-tiedostossa olevalla *session.gc_maxlifetime*-arvolla voidaan määritellä kuinka monta sekuntia rekisteröity istunto on voimassa, ennen kuin se hylätään. Jokaisella GateKeeper-ohjelmiston näkyvällä sivulla on ”Poistu”-nappi, jolla voimassaoleva istunto voidaan kuitenkin päättää ennen määräajan täyttymistä. Sitä käyttämällä suoritetaan *lopetaIstunto()*-funktio, joka poistaa istunnon globaalimuuttujat, sekä suorittaa *session_destroy()*-funktion. Poistu-nappia käyttämällä torjutaan mahdollisuus kopioida selaimen otsikkorivillä oleva *PHPSESSID*-tunnus ja käyttää ohjelmistoa liittämällä se uuteen selainikkunaan.

7.2 Ohjelman prosessikuvaukset

GateKeeper-ohjelman keskeisimmät toiminnot ovat käyttäjien lisääminen, salasanojen vaihto ja oikeuksien myöntäminen Mambo, Moodle ja Subversion-ohjelmille. Ohjelman toteutuksen kannalta on haaste, että useat toiminnot edellyttivät käyttäjätietojen päivittämistä useaan eri järjestelmään. GateKeeper-ohjelmisto käsittelee GateKeeper-tietokannan lisäksi Mambo-, Moodle- ja Subversion-tietokantoja. Toteutuksen lähtökohtana on, että GateKeeper-tietokantaan on tallennettu kaikki tiedot ja ohjelmien omiin tietokantoihin kopioidaan tietoa vain tarvittaessa (kuva 7). Mikäli tiedoissa on ristiriitoja, muut tietokannat päivitetään GateKeeper-tietokantaa vastaaviksi.



Kuva 7. GateKeeper-ohjelma noudattaa kolmikerrosarkkitehtuuria. Ohjelman toimintaperiaatteen mukaisesti käyttäjätiedot kopioidaan tarvittaessa muiden ohjelmien tietokantoihin GateKeeper-tietokannasta.

Ohjelma koostuu moduuleista, joiden toiminnot ovat toisistaan riippuvaisia. Esimerkiksi salasanan vaihtaminen ja käyttöoikeuden myöntäminen tietylle ohjelmalle edellyttävät, että käyttäjätunnus esiintyy jo ohjelman omassa tietokannassa. Tämä käy ilmi esimerkiksi SQL-lauseesta:

$kysely_mambo= "UPDATE mambo_users SET password = '$salasana',
WHERE username = '$kayttaja'"

Lause tallentaa salasanan Mambon omaan tietokantaan, jolloin käyttäjälle myönnetään oikeus käyttää ohjelmaa. Käyttäjän salasanaa ei voida kuitenkaan tallentaa ohjelmien omiin tietokantoihin aina kun käyttäjän salasanaa vaihdetaan. Tallennus voisi antaa käyttäjälle tahattomasti oikeuden käyttää ohjelmia. Käyttöoikeus toteutuu, mikäli ohjelman omasta tietokannasta löytyy vähintään käyttäjätunnus ja salasana. Sen vuoksi ennen salasanan tallennusta tarkistetaan GateKeeper-tietokannasta, onko käyttäjälle myönnetty oikeus käyttää kyseistä ohjelmaa. Muuttunut salasana tallennetaan ohjelman omaan tietokantaan vain, mikäli käyttöoikeus on voimassa. Luonnollisesti tarkistus tehdään jokaisen ohjelman osalta erikseen ja jokaisessa kohdassa, jossa käyttäjätietoja muutetaan.

7.2.1 Index_admin

Index_admin.php on sivu, jolle pääkäyttäjä ohjataan onnistuneen kirjautumisen jälkeen. Index_admin.php-sivulla pääkäyttäjä voi vaihtaa oman sekä kenen tahansa muun käyttäjän salasanan. Sivulle on sijoitettu myös linkit ohjelmiston muille sivuille. Index_admin.php- sivun ohjelmakoodi on esitetty lyhennettynä liitteessä 2.

Salasanan vaihtaminen tapahtuu valitsemalla ensimmäiseksi hakuruudusta käyttäjätunnus. Tunnukset saadaan hakuruutuun GateKeeper-tietokannasta SQL-kyselyn avulla. Pääkäyttäjä voi määrittää itse käyttäjälle uuden salasanan kahteen seuraavaan password-tyyppiseen tekstikenttään, tai antaa ohjelman arpoa uuden, satunnaisen salasanan. Arvottua salasanaa on käytännössä mahdotonta arvata ja sitä käytetään vain, jos käyttäjä halutaan sulkea pois järjestelmästä. Mikäli käyttäjän salasanaksi tallennetaan arvottu salasana, ei käyttäjää enää seuraavilla hakukerroilla näy hakuruuduissa.

Syötetyt salasanat tarkistetaan *KentanTarkistus()*-funktion avulla. Mikäli annetut salasanat täsmäävät, tarkistetaan käyttäjälle myönnetyt ohjelmien käyttöoikeudet. Tarkistus tehdään SQL-kyselyn avulla. Uusi salasana päivitetään vain niihin ohjelmiin, joiden käyttöoikeus on myönnetty. Salasana tallennetaan GateKeeper-, Mambo- ja Moodle-tietokantoihin md5-salausfunktion avulla. Subversion-ohjelma käyttää salasanojen tallennukseen HTTP-autentikaation mukaista salasanatiedostoa. Uusi salasana päivitetään salattuna niiden

tietovarastojen salasanatiedostoihin, joihin käyttäjälle on myönnetty käyttöoikeus. Lopuksi tehty salasanan muutos päivitetään historiatietoihin.

7.2.2 Index_user

Index_user on ainoa sivu, jolle tavallisella käyttäjällä on pääsy kirjautumissivun lisäksi. Sivulla on kaksi password-tyyppistä tekstikenttää, joihin käyttäjä voi kirjoittaa uuden salasanansa. Lomakkeella lähetettyjen salasanojen pituus ja merkistö tarkistetaan *kentanTarkistus()*-funktiolla. Mikäli salasanat täsmäävät ja läpäisevät tarkistuksen, tarkistetaan käyttäjälle myönnetyt ohjelmien käyttöoikeudet. Tarkistus tehdään SQL-kyselyn avulla. Uusi salasana päivitetään vain niihin ohjelmiin ja tiedostoihin, joiden käyttöoikeus on myönnetty. Tehty salasanan muutos päivitetään historiatietoihin.

7.2.3 AddUser

Adduser.php on ainoastaan pääkäyttäjille näkyvä sivu. Sivulla määritetään uudelle käyttäjälle tunnus, salasana sekä annetaan käyttäjän muita tietoja. Käyttäjän lisäystä varten käyttäjätunnus-kenttään annetaan yksilöllinen, aakkosnumeerinen, neljästä viiteentoista merkkiin pitkä merkkijono. Mikäli tietokannassa on jo samanniminen tunnus, MySQL-tiedonhallintajärjestelmä antaa siitä virheilmoituksen, eikä tallennusta suoriteta. Uuden käyttäjän salasanat syötetään kahteen eri kenttään. GateKeeper tarkastaa syötteiden samankaltaisuuden sekä pituuden. Mikäli tunnus ja salasanat läpäisevät tarkistuksen, ne tallennetaan GateKeeper-tietokantaan. Salasana tallennetaan GateKeeper-tietokantaan md5-funktiolla salattuna. Sen lisäksi salasanasta tehdään HTTP-autentikaation mukainen salattu muunnos, joka tallennetaan myös GateKeeper-tietokantaan.

Käyttäjän lisäyksen yhteydessä uusi käyttäjätunnus lisätään myös Mambo- ja Moodle-tietokantoihin. Käyttäjän oman salasanan sijasta näihin viedään kuitenkin arvottu salasana, jota käyttäjä ei pysty arvaamaan. Tällä menettelyllä varmistetaan, että Mambo- ja Moodle-ohjelmia voidaan käyttää vasta, kun niiden käyttöoikeus on myönnetty GateKeeper-ohjelmassa erikseen. Subversionin vastaava menettely eroaa siten, että käyttäjä lisätään salasanatiedostoihin vasta käyttöoikeuden myöntämishetkellä.

Uuden käyttäjän lisäyksen jälkeen adduser-sivulla voidaan tallentaa myös käyttäjää koskevia muita tietoja. Kun uusi käyttäjä valitaan luetteloruudusta, sivulle ladataan lomake

käyttäjän tietojen syöttämistä varten. Tietokantaan voidaan tallentaa käyttäjän etu- ja sukunimi, sähköpostiosoite, ICQ- ja puhelinnumerot sekä osoitetiedot. Ennen tallennusta ohjelma tarkistaa, että sähköpostiosoite ja numerotiedot on annettu oikeassa muodossa.

7.2.4 UserRights

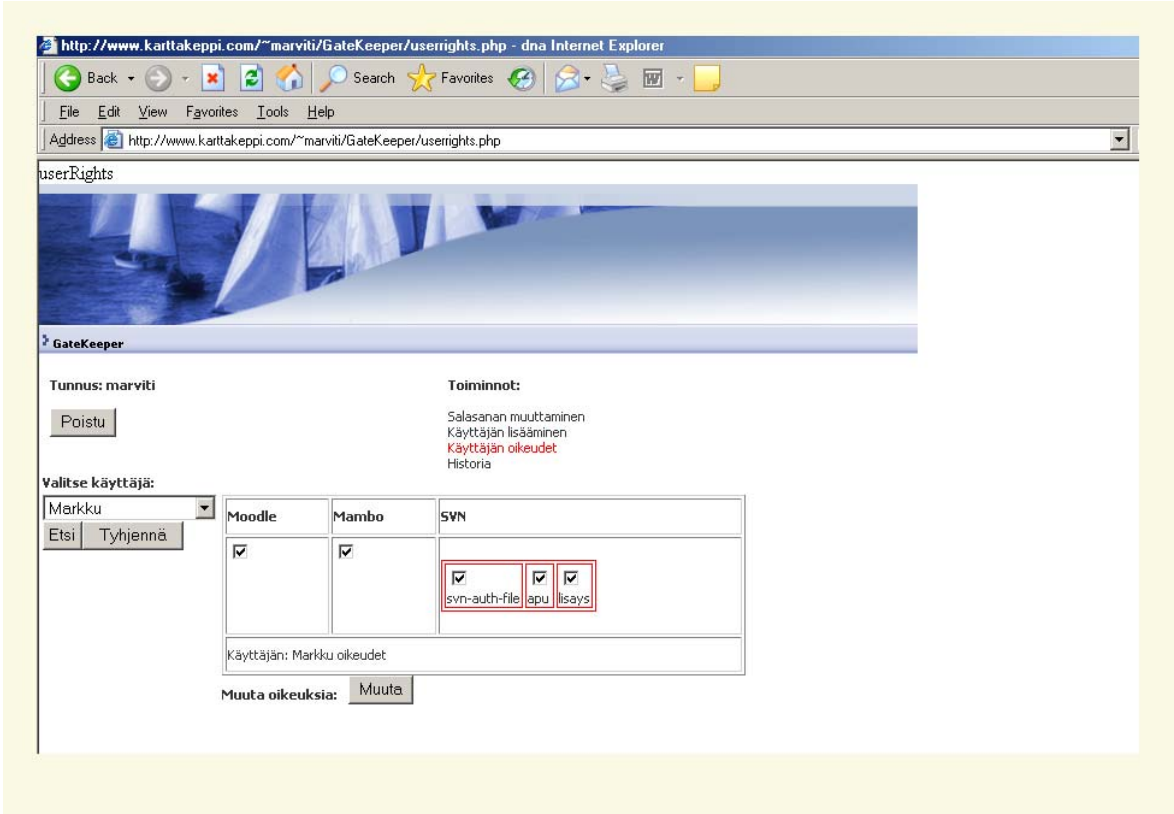
Userrights.php on GateKeeper-ohjelman keskeisin osa, jossa eri ohjelmien käyttöoikeuksia tarkastellaan, myönnetään tai evätään. Sivu näkyy luonnollisesti vain pääkäyttäjille. Heillä on oikeus muuttaa kaikkien, myös muiden pääkäyttäjien ohjelmia koskevia oikeuksia kuvassa 8 esitetyn käyttöliittymän avulla.

Oikeuksien tarkastelu alkaa valitsemalla käyttäjä luetteloruudusta. Ohjelma listaa ohjelmien oikeudet taulukkomuodossa, jossa jokaista erillistä oikeutta vastaa valintaruutu. Valintaruutu näkyy valittuna, mikäli kyseinen oikeus on myönnetty. Tiedot valintaruutuihin on saatu oikeudet-taulusta SQL-kyselyllä:

```
$kysely_oikeudet = "SELECT oikeudet.oikeus_moodle, oikeu-
det.oikeus_mambo FROM oikeudet,tunnistus WHERE tunnis-
tus.tunnistus_kayttaja = oikeudet.oikeus_kayttaja AND oikeu-
det.oikeus_kayttaja = '$kayttaja'";
```

Kyselyn tulos haetaan *MySQL_fetch_array()*-funktiolla assosiatiiviseen taulukkoon, joka läpikäydään *foreach()*-funktiolla. Jokaista saraketta vastaa valintaruutu, jolle annetaan oma nimi. Siten valintaruutuja voidaan käsitellä, kuten bool-tyyppisiä muuttujia:

```
while ($rivi = MySQL_fetch_array($result_oikeudet, MYSQL_ASSOC)) {
        $i=1;
        foreach ($rivi as $sarakkeen_arvo) {
        if ( $sarakkeen_arvo == 1) $rasti = "checked";
        else $rasti ="";
        echo "<td"><input type='checkbox' name=oikeus$i $rasti>";
        $i++;
        echo "</td>";
        }
```



Kuva 8. GateKeeper-ohjelman userRights.php-sivulla lisätään ja poistetaan ohjelmien oikeuksia

Valintaruutujen tilaa voidaan muuttaa. Niiden arvot tallennetaan tietokannan oikeudet-tauluun tallenna-painikkeella. Kun valintaruutu valitaan eli käyttäjälle annetaan ohjelman käyttöoikeus, käyttäjän salasana päivitetään myös kyseisen ohjelman tietokantaan. Mambon ja Moodlen käyttöoikeus toteutuu, kun niiden omaan MySQL-tietokantaan on talletettu käyttäjätunnus ja salasana. Mikäli valintaruutu jätetään tyhjäksi, oikean salasanan sijasta valintaruutua vastaavan ohjelman omaan tietokantaan tallennetaan arvottu salasana. Tällöin kyseinen käyttäjä ei pysty enää käyttämään ohjelmaa.

Subversionin käyttäjienhallintamekanismi perustuu salasanatiedostoihin, joista jokainen vastaa tiettyä tietovarastoa. Kaikki Subversionin salasanatiedostot on tallennettu samaan hakemistoon. Käyttäjän oikeuksien tarkastelun yhteydessä, GateKeeper lukee läpi koko hakemiston ja kaikki siinä olevat tiedostot. Jos käyttäjälle on myönnetty tietovaraston käyttöoikeus, sitä vastaavasta salasanatiedostosta löytyy rivi:

tunnus:salasana_salatussa_muodossa

Hakemisto käydään läpi *tarkastaSVNTiedostot()*-funktiolla, jolle annetaan parametreinä GateKeeper-tietokannasta haettu käyttäjän nimi ja salasana. Funktio avaa annetussa polussa olevan hakemiston ja antaa siinä olevat salasanatiedostot yksi kerrallaan

parametrinä *lueSVNTiedostot()*-funktion käsiteltäviksi. Tämä funktio palauttaa arvon *true*, mikäli ennen mainittu käyttäjätunnuksen ja salasanan sisältävä rivi löytyy tiedostosta.

```
function tarkistaSVNTiedostot($kayttaja,$kayttaja_salasana_svn) {
        $i  = 1;
        $polku = "AuthFiles";
        $this->kayttaja = $kayttaja;
        $this->$kayttaja_salasana_svn =$kayttaja_salasana_svn;

        $hakemistot = array();
        $tiedostot = array();
        $hakemisto = opendir($polku);
        while (($kahva = readdir($hakemisto)) !== false) {
        if ($kahva != "." && $kahva != ".." &&  substr($kahva, -4) != ".php"){

        if ($kahva !== (is_dir($kahva))) {
        if(lueSVNTiedosto($kahva,$kayttaja,$kayttaja_salasana_svn) == true)
                $rasti = "checked";
                else $rasti ="";
        echo "<input type ='checkbox' name =kahva$i $rasti><br>$kahva";
        $i++;
        } } }
        closedir($hakemisto);
        }
```

LueSVNTiedosto()-funktio jakaa tiedostossa olevan merkkijonon ensin välilyönnin perusteella riveihin. Tämän jälkeen se erottaa kaksoispisteen perusteella jokaisen rivin alku- ja loppuosan. Alkuosaa verrataan käyttäjätunnukseen ja loppuosaan käyttäjän salasanaan:

```
function lueSVNTiedosto(){
.
.
.
$jonot = explode(':', $rivit[$i]);
$user=$jonot[0];
$pass=$jonot[1];
if((strcmp($nimi,$user) == 0) && (strcmp($salasana,$pass) ==0)){
                $loydetty++;
                }
if ($loydetty ==1) return true;
}
```

Kahden kuvatun funktion avulla saadaan aikaan jokaista tietovarastoa vastaava valintaruutu, joka näkyy valittuna, mikäli käyttäjälle on myönnetty tietovaraston käyttöoikeus. Kun valintaruudun arvoa muutetaan, eli oikeus tietovaraston käyttämiseen myönnetään tai lisätään, käytetään *paivitaTiedosto()*-funktiota. Funktiolle on annettu parametreina salasanatiedoston nimi, sekä käyttäjän uusi salasana. Ensimmäisenä funktiossa tarkistetaan löytyykö käyttäjätunnus jo tiedostosta. Mikäli käyttäjää ei ole vielä lisätty tiedostoon, se

lisätään tässä vaiheessa. Jos käyttäjän oikeus evätään, korvataan vanha salasana arvotulla salasanalla. Jos taas käyttäjälle myönnetään oikeus, korvataan vanha, mahdollisesti arvottu salasana käyttäjän oikealla salasanalla. Tätä varten tiedosto avataan ”w”-moo-dissa, löydetty rivi korvataan uudella ja koko tiedosto kirjoitetaan uudestaan:

```
// Jos käyttäjä löytyi,vanha salasana korvatataan uudella:
if ($loydetty>0){
        $kahva = fopen( $tiedosto,'w')or die("Tiedostovirhe");
        $uusiteksti = str_replace($loydetty_rivi, "$nimi:$salasana",
        $teksti);
        fwrite($kahva, $uusiteksti);
        fclose($kahva);
        }
```

Mikäli käyttäjätunnusta ei löytynyt tiedostosta, uusi käyttäjätunnus ja salasana lisätään tiedoston loppuun. Tällöin tiedosto avataan ”a”-moodissa:

```
//Jos käyttäjän nimeä ei löytynyt, tiedostoon lisätään käyttäjä ja salasana:
if ($loydetty==0)
        {
        $kahva = fopen( $tiedosto,'a')or die("Tiedostovirhe");
        // Vaihtaa rivin
        fwrite($kahva,"\n");
        //Kirjoittaa käyttäjätunnuksen:
        fwrite($kahva, $nimi);
        // Kirjoittaa kaksoispisteen:
        fwrite($kahva,":");
        // Kirjoittaa salasanan:
        fwrite($kahva,$salasana);
        fclose($kahva);
        }
```

8 JÄRJESTELMÄN TESTAUS, LAADUNVARMISTUS JA YLLÄPITO

8.1 Testaussuunnitelma

Testaussuunnitelman mukaisesti ennen ohjelman varsinaista käyttöönottoa sille suoritetaan moduuli-, integrointi-, sekä järjestelmätestaus. Testitapaukset perustuvat sekä toiminnallisiin että ei-toiminnallisiin asiakasvaatimuksiin. Testaus tapahtuu sitä varten rakennetussa testiympäristössä.

Moduulitestaus on kuulunut kiinteästi GateKeeper-ohjelman toteutukseen. Monesti moduulin suunnittelu, ohjelmointi ja testaus ovat edenneet jopa rinnakkain. Voidaan sanoa, että moduuleihin on lisätty toimintoja sitä mukaa, kun edelliset toiminnot on todettu toimiviksi. Ohjelman koodaus on tapahtunut teknisen määrittelyn pohjalta.

Onnistuneen moduulitestauksen jälkeen moduulien yhteensopivuutta on tarkasteltu integrointitestauksen avulla. Integrointitestaus on edennyt myös ohjelmoinnin ja moduulitestauksen rinnalla. Integrointitestauksen onnistumisen edellytyksenä on ollut toimiva testiympäristö.

Järjestelmätestauksessa testitapaukset ja tulokset perustuvat määrittelydokumentaatioon /2,s.288/. Myös järjestelmätestaus on suoritettu testiympäristössä. Järjestelmätestaukseen on pyritty löytämään projektin ulkopuolisia henkilöitä riippumattomuuden takaamiseksi. Testaajille on annettu mahdollisuus ja ohjeistus testiympäristön käyttöön, sekä testitapaukset sisältävä testauspöytäkirja. Testaus on ollut pääosin mustalaatikkotyyppistä käytettävyystestausta. Ohjelmaa on muutettu testaustulosten perusteella. Testiohjelmassa seuraavana tarkistuskohteena oli ohjelman tietoturva. Sen päätyttyä testauksen oletetaan jatkuvan asiakkaan suorittamana beta-testauksena.

8.2 Testiympäristö

Sekä moduuli- että integrointitestauksen onnistumisen edellytys on toimiva testiympäristö. Keskeneräistä ohjelmaa ei voida testata varsinaisessa tuotantoympäristössä, jossa häiriöt eivät ole toivottuja. Tästä syystä jo projektin alkuvaiheessa pyrittiin rakentamaan

mahdollisimman paljon varsinaista tuotantoympäristöä muistuttava testiympäristö, jossa ohjelmaa voidaan testata täysin vapaasti.

Testiympäristön rakentaminen aloitettiin palvelimen asennuksella. Palvelimen käyttöjärjestelmäksi valittiin Linux Mandrake 10.1 -distribuutio. Asennuspakettiin valittiin mukaan Apache 2.0 palvelinohjelmisto, PHP-tuki sekä MySQL-tiedonhallintajärjestelmä. Tämän jälkeen palvelin olikin valmis liitettäväksi verkkoon. Jonkin ajan kuluttua palvelimella havaittiin kuitenkin huomattavan paljon liikennettä. *Netstat*-komento osoitti palvelupyyntöjen tulevan varsin epäilyttävistä osoitteista. Tässä vaiheessa oli aiheellista perehtyä palvelimen tietoturvallisuuteen hieman paremmin. *Register_globals*-asetuksen muuttaminen off-tilaan vähensi tässä tapauksessa merkittävästi epätoivottua liikennettä. Palomuuriasetusten tarkistaminen lisäsi osaltaan turvallisuudentunnetta. Näiden toimenpiteiden lisäksi Apachen määrittelytiedostoon lisättiin vielä *mod_security.so*-moduuli, jonka tehtävä on suojata web-sovelluksia hyökkäyksiä vastaan. Tämän jälkeen palvelin vaikutti olevan jo melko hyvin suojattu, eikä epätoivottua liikennettä enää näyttänyt enää esiintyvän.

Mambo, Moodle ja Subversion -ohjelmien asennus palvelimelle lisäsi tietoa näiden ohjelmien ominaisuuksista. GateKeeper-ohjelman kehitystyön kannalta oli välttämätöntä selvittää joitakin perusasioita näiden ohjelmien käyttäjienhallinnasta ja toiminnasta. Projektissa ei kuitenkaan ollut mahdollisuuksia ohjelmien kaikkien ominaisuuksien opiskeluun, joten siinä keskityttiin ainoastaan ohjelmien asennukseen ja käyttäjien tunnistustapaan. Lopulta kaikkien kolmen ohjelman käyttäjätietoja pystyttiin hallinnoimaan GateKeeperin avulla.

8.3 Laadunvarmistus

GateKeeper-ohjelman laadunvarmistus on pyritty varmistamaan vaiheistamalla ohjelman kehitystyö, sekä järjestämällä riittävästi tarkastuksia jokaisessa vaiheessa. Ohjelman kehitystyössä on tukeuduttu Ilkka Haikalan ja Jukka Märijärven Ohjelmistotuotantoteokseen. Noudattamalla teoksessa kuvattuja toimintaprosesseja on pyritty varmistamaan tuotteen sekä tuotetta valmistavien prosessien laatu.

Asiakas eli Karttakeppi.comin edustaja on ollut mukana tuotteen kehitystyössä alusta lähtien sen jokaisessa vaiheessa. Ohjelman ohjelmointivaiheen lopussa järjestetty koodin katselmus toi esiin tiettyjä seikkoja, joihin asiakas toivoi kiinnitettävän enemmän huomiota. Asiakas painotti tyyliseikkojen merkitystä ohjelman koodauksessa. Ohjelman ylläpidon ja

kehityksen kannalta olisi tärkeää noudattaa annettuja tyylisääntöjä ja -suosituksia. Asiakas suositteli erillisten tyylisivujen käyttämistä. Asiakkaan ohjelmointityyliin liittyvät toivomukset pystytään toteuttamaan tässä projektissa osittain. Ennen ohjelman käyttöönottoa ohjelmakoodi on käyty läpi ja muuttujien sekä funktioiden nimeäminen on muutettu tyylioppaan ohjeiden mukaiseksi. Ohjelman ulkoasua ei annetulla aikataululla pystytä täysin erottamaan sen rakenteesta, mutta näihin seikkoihin tullaan kiinnittämään huomiota tulevaisuudessa.

8.4 Ohjelman integrointi ja ylläpito

GateKeeper-ohjelma on kehitetty määrättyyn ympäristöön toimimaan yhdessä nimettyjen ohjelmien kanssa. Ohjelman kehitystyössä on kuitenkin otettu huomioon myös uusien ohjelmien integroiminen ohjelmaan. Tässä luvussa kuvataan, miten GateKeeper-ohjelma otetaan käyttöön uudessa ympäristössä ja miten uudet ohjelmat voidaan integroida siihen.

SimpleAuth-mekanismin mukaisesti GateKeeper-ohjelma tarkistaa kirjautumisen yhteydessä käyttäjän tunnuksen, salasanan ja roolin GateKeeper-tietokannasta. Tätä varten ohjelmalle on ensimmäisenä kerrottava GateKeeper-tietokannan sijainti, nimi, käyttäjätunnus sekä salasana. Mikäli tietokantaa ei vielä ole olemassa, se täytyy tässä vaiheessa luoda. Tyhjä GateKeeper-tietokanta voidaan kopioida ja siirtää toiselle palvelimelle esimerkiksi testiympäristöstä.

MySQL:n hallintaohjelmalla on luotava GateKeeper-ohjelmalle myönnettävä käyttäjätunnus, jolla on vähintään *select, insert,* ja *update* -oikeudet. Käyttäjän luonti tapahtuu esimerkiksi komennoilla:

```
MySQL> grant select, insert, update on gatekeeper.* to tunnus@localhost
identified by ‘salasana’;
MySQL>flush privileges;
```

Ensimmäinen varsinainen käyttäjätunnus ja salasana GateKeeper-tietokantaan täytyy luoda käsin MySQL-ohjelman avulla. Käyttäjätunnuksella täytyy olla *Administrator-rooli,* eli myös roolit on määriteltävä sitä ennen roolit-tauluun. Käyttäjän salasana esiintyy tietokannassa salattuna. Ensimmäinen käyttäjä voidaan siten lisätä tunnistus-tauluun esimerkiksi komennolla:

MySQL> INSERT INTO tunnistus (tunnistus_kayttaja, tunnistus_rooli, tunnistus_salasana) VALUES ('Käyttäjätunnus', 1, md5('Salasana'));

Edellytyksenä Mambon ja Moodlen toiminnalle yhdessä GateKeeper-käyttäjienhallintaohjelman kanssa on, että molemmilla ohjelmilla on oma MySQL-tyyppinen tietokanta. Etenkin Moodlen asennuksen yhteydessä tarjolla on myös muita vaihtoehtoja. Mambo- ja Moodle-ohjelmien tietokannat voivat sijaita samalla tai jollakin toisella palvelimella kuin GateKeeper-tietokanta. Luotettavuuden kannalta on kuitenkin etu, jos tietokannat sijaitsevat samalla palvelimella. GateKeeper-ohjelma tarvitsee käyttäjä-tunnuksen, jolla on vähintään *select, insert,* ja *update* -oikeudet molempien ohjelmien tietokantoihin. GateKeeper luo yhteyden tietokantoihin *functions.php*-moduulissa määritetyillä *yhdistaMoodle()*- ja *yhdistaMambo()*-funktioilla. Molemmat funktiot käyttävät *MySQL_pconnect()*-funktiota, jolle on kerrottava palvelimen osoite, tietokannan nimi, käyttäjätunnus sekä salasana.

Subversion-ohjelman integroiminen edellyttää, että kaikki salasanatiedostot on tallennettu samaan hakemistoon, jossa ohjelmalla on luku- ja kirjoitusoikeudet. Hakemiston polku määritellään *tarkistaSVNTiedostot()*-funktiossa. Ohjelma tallentaa salasana-tiedostojen nimet GateKeeper-tietokantaan. *TarkistaSVNTiedostot()*-funktio välittää tiedostojen nimet *UserRights.php*-moduulille, joka päivittää SQL-kyselyn avulla tiedot tietokantaan. UserRights.php-sivulle on rakennettu mekanismi kymmenelle salasanatiedostolle. Sivun ohjelmakoodissa joudutaan kommentoimaan kaksi eri kohtaa, jotka riippuvat salasanatiedostojen määrästä. Mikäli salasanatiedostoja on vähemmän kuin kymmenen kappaletta, antaa ohjelma virheilmoituksen ilman kommentointia yrittäessään tallentaa tyhjää arvoa tietokantaan.

Uutta ohjelmaa lisättäessä joudutaan muutoksia tekemään sekä tietokantaan että ohjelmaan. Tietokannan oikeudet-taulussa jokaista ohjelmaa vastaa bool-tyyppinen muuttuja, jonka arvo riippuu siitä onko käyttäjälle myönnetty käyttöoikeus. UserRights-moduulissa oikeudet-taulu luetaan ja valintaruudut näytetään valittuina saadun tuloksen mukaisesti. Uudelle ohjelmalle syntyy automaattisesti valintaruutu. Vain ohjelman nimi joudutaan lisäämään taulukon otsikkoriville.

9 YHTEENVETO

Nykyään tiedotusvälineissä puhutaan jatkuvasti kansainvälisestä kilpailusta ja Suomen kilpailukyvystä kansainvälisillä markkinoilla. Viime aikojen kehitys on valitettavasti osoittanut, että menestyäkseen globaalissa kilpailutilanteessa monien yritysten on ollut välttämätöntä siirtää tuotanto lähemmäksi markkinoita, halvempien tuotantokustannusten maihin. Tämän tosiseikan eteen ovat joutuneet erityisesti aineellisia hyödykkeitä valmistavat yritykset. Niinpä Suomessa onkin pyritty löytämään uusia, korvaavia tuotannon aloja, jotka voisivat toimia kilpailukykyisesti maassamme. Päättäjien taholta on annettu ymmärtää, että Suomen mahdollisuudet tulevaisuudessa liittyvät korkean teknologian tuotteisiin, joissa tärkein tekijä menestymisessä kansainvälisillä markkinoilla on osaaminen. Työpaikkojen syntymistä toivotaan erityisesti suunnittelu- ja tuotekehitystehtäviin. Tulevaisuuden turvaamiseksi Suomessa onkin määrätietoisesti panostettu koulutukseen ja tuettu uusia kasvualoja.

Ohjelmistoala kuuluu varmasti niiden alojen joukkoon, jossa maantieteellisillä rajoilla on toiminnan kannalta vähiten merkitystä. Teknisesti samanlainen ohjelmistotuote voidaan valmistaa missä päin maailmaa tahansa ja ottaa käyttöön välittömästi sen jälkeen jossakin muualla. Menestyäkseen Suomessa ohjelmistotuotannon on siis oltava tehokkaampaa ja laadukkaampaa kuin muualla. Koska ohjelmistotuotantoon liittyvä tekniikka ja työvälineet kehittyvät jatkuvasti huimaa vauhtia, on tällä alalla työskentelevien välttämätöntä seurata kehitystä ja pidettävä osaaminen aina ajan tasalla. Vanhentuvilla työmenetelmillä ja -välineillä ei ole mahdollista saavuttaa tarvittavia kilpailuetuja.

Kuten tiedetään, ohjelmistoalalla ei synny tuotteita, joita olisi mahdollista kosketella käsin ja joiden tuotannolla olisi pitkät perinteet. Ohjelmistotuotanto on verrattain nuori tuotannon ala, jolla edelleen etsitään parhaita tuotanto- ja kehitysmenetelmiä. Ohjelmistotyölle tyypillisiä ominaisuuksia ovat monimutkaisuus, näkymättömyys, muunnettavuus, ainutkertaisuus, skaalautumattomuus sekä epäjatkuvuus /2, s.29-30/. Tietyn ohjelmistoprojektin onnistumista on siis vaikea arvioida, koska sille on vaikea löytää selkeää vertailukohtaa. Ohjelmistotuotteiden valmistusta varten on kuitenkin kehitetty useita malleja ja menetelmiä, jotka tähtäävät niiden laadukkaaseen kehitykseen ja lopputulokseen. Lähtökohtana näille malleille on asiakaslähtöisyys, eli ohjelmistokehityksen tavoitteena on aina tuottaa asiakkaan tarpeita vastaava ohjelma. Vähitellen ohjelmistotuotannossa onkin yleisesti alettu hyväksyä tietyt toimintamallit, joita voidaan

käyttää hyväksi sellaisenaan, tai niistä poiketen. Malleille tyypillinen ominaisuus on vaiheistus, eli ohjelmistokehityksen eteneminen vaiheesta toiseen. Jokaiseen vaiheeseen liittyy tukitoimintoja, kuten laadunvarmistus, tuotteenhallinta ja dokumentointi. Näiden menetelmien uskotaan tuovan kehitykseen tehokkuutta ja selkeyttävän toimintatapoja. Toimintamallien mukaisesti valmista tuotetta pystytään testaamaan ja vertaamaan asiakasvaatimuksiin nähden. Ohjelmistoprojektin onnistumista onkin siis mahdollista arvioida sillä perusteella, miten se täyttää sille asetetut vaatimukset.

Tyypillisesti ohjelmistoista löytyy aina virheitä, jotka aiheuttavat tarpeen muuttaa tai kehittää ohjelmaa edelleen. Mikäli ohjelmalle on ylipäätään tarvetta, sen kehitys yleensä jatkuu tavalla tai toisella. Olennaista kuitenkin on, että tuotantoympäristössä käyttöönotetusta ohjelmasta on oltava asiakkaalle enemmän hyötyä kuin haittaa. Nykyisessä tuotantoympäristössä myös vaarantunut tietoturva on laskettava haitaksi. Tämän vuoksi ohjelmien kehityksessä on aina otettava huomioon myös tieto-turvanäkökohdat. Tietoturvan osalta keskeneräistä tuotetta ei ole syytä ottaa käyttöön ennen perusteellista testausta ja epäkohtien korjaamista.

Tämän Insinöörityön loppuvaiheessa GateKeeper-ohjelman ensimmäinen versio on valmis. Kuten Haikalan ja Märijärven Ohjelmistotuotanto-kirjassa todetaan, Evo-malli muodostuu sarjasta vesiputouksia, joista jokaisen tuloksena on uusilla ominaisuuksilla kasvatettu järjestelmä /2, s.41/. GateKeeper-projektin tavoitteena on ollut tuottaa pienimuotoinen, laadukas, asiakkaalle hyödyllinen ohjelma alussa mainitun laajemman projektin tietoturvallisuutta vaarantamatta. Laadukkuuteen on pyritty noudattamalla ohjelmistotuotannon kestäviä menetelmiä kuten vaiheistusta, tarkistuksia, sekä dokumentointia. Ilmeisesti tavoite on kaikkien osapuolten mielestä saavutettu. Joka tapauksessa projekti on ollut oppimistarkoituksessa tekijälle erittäin hyödyllinen. Toivottavasti ohjelman tuotekehitys tulee edelleen jatkumaan ja lopulta voidaan todeta, että sen elinkaari on noudattanut todellista Evo-mallia.

VIITELUETTELO

/1/ Ficora, viestintävirasto. [verkkodokumentti]. *Internetin tietoturva*. 1.9.2004. [viitattu 26.12.2005]. Saatavissa: http://www.ficora.fi/suomi/tietoturva/tietoverkot.htm

/2/ Haikala, Ilkka - Märijärvi, Jukka. *Ohjelmistotuotanto.*Helsinki: Satku - Kauppakaari. 2002.

/3/ Tampereen Teknillinen Yliopisto [verkkodokumentti]. *Menetelmät ja työohjeet*. Tampereen Teknillinen Yliopisto. 27.8.1998. [viitattu 20.1.2006]. TTY>Menetelmät ja työohjeet>Dokumentointi. http://www.cs.tut.fi/cgi-bin/laatu/sivuhaku.pl?nk_no=0&nk_id=0

/4/ Valkealahti, Markku. *Toiminnallinen Määrittely, GateKeeper.* 12.11.2005. [viitattu 14.1.2006]. Saatavissa: http://www.karttakeppi.com/~marviti/toiminnallinenmaarittely.pdf

/5/ Vitikainen, Markku. *Projektisuunnitelma, Gatekeeper.* 8.1.2006. [viitattu 14.1.2006]. Saatavissa: http://cs.stadia.fi/~0103080/inssityo/ProjektiSuunnitelma.pdf

/6/ Vitikainen, Markku. *Tekninen Määrittely, GateKeeper.* 15.1.2006. [viitattu 20.1.2006]. Saatavissa: http://cs.stadia.fi/~0103080/inssityo/TekninenMaarittely.pdf

/7/ Vitikainen, Markku. *Testaussuunnitelma, GateKeeper.*15.1.2006. [viitattu 15.2.2006]. http://www.tml.tkk.fi/Opinnot/T-110.470/2004/20041122.pdf

/8/ Nikunen, Erja. *Projektinhallinta,* Luentokalvot. Ohjelmistotuotannon perusteet. Helsingin Ammattikorkeakoulu. Helsinki. Kevät 2003.

/9/ Silander, Simo - Peltomäki, Juha. *Java 2, Ohjelmoinnin peruskirja.* Helsinki. Docendo Finland. 2002.

/10/ Wikipedia [verkkodokumentti] *Overview of a three-tier application.* 31.8.2005 [viitattu 15.2.2006]. Saatavissa*:* http://en.wikipedia.org/wiki/Image:Overview_of_a_three-tier_application.png

/11/ Tampereen Teknillinen Yliopisto.[verkkodokumentti] *Verkkopalvelu-arkkitehtuuri*. 25.1.2005. [viitattu 15.2.2006] Saatavissa: http://matwww.ee.tut.fi/hmopetus/hm-ohj/2005/pruju/hmohj05-037-046-4-sivulla.pdf

/12/ Hyrskykari, Aulikki, *Vuorovaikutteisuutta www-sivuille Javascriptin avulla*. [verkkodokumentti]. Tampereen Yliopiston Tietojenkäsittely-tieteiden laitos. [viitattu 15.2.2006] Saatavissa: http://www.cs.uta.fi/~ah/javascript/johdanto.html

/13/ Heinisuo, Rami, *PHP ja MySQL.* Helsinki. Talentum.

/14/ Steffensen, Jan, *Introduktion til HTML.* Malmö. Egmont Richter AB. 1999.

/15/ Hancoc, Jez. [verkkodokumentti]. *PHP Simple Auhentication Using MySQL and PHP sessions.* 1.3.2003 [viitattu 30.1.2006]. Saatavissa: http://simpleauth.munk.nu/

/16/ The PHP Group. [verkkodokumentti]. *Session Handling Functions* 14.12.2005 [viitattu 30.1.2006]. Saatavissa: http://www.php.netmanual/fi/ref.session.php

/17/ Rajamäki, Juhani. [verkkodokumentti]. *Tiedonhallinnan perusteet, kurssimoniste.* 1.5.2000. Helsingin Ammattikorkeakoulu. [viitattu 15.1.2006]. Saatavissa: http://cs.stadia.fi/~lehtonen/TiTe/Tietokannat/thperusteet.doc

/18/ The World Wide Web Consortium (W3C). [verkkodokumentti].The *World Wide Web Security FAQ.* 23.2.2003. [viitattu 5.3.2006]. Saatavissa: http://www.w3.org/Security/Faq/

/19/ Builder.com. [verkkodokumentti]. *A comparison of Web service security techniques.* 6.6.2002. [viitattu 2.4.2006]. Saatavissa: http://builder.com.com/

/20/ Parse-Error.com. [verkkodokumentti]. *Lomakkeita tietoturvallisesti PHP:n avulla.* [viitattu 8.3.2006] Saatavissa: http://www.parse-error.com/php_internet_ohjelmointi.php

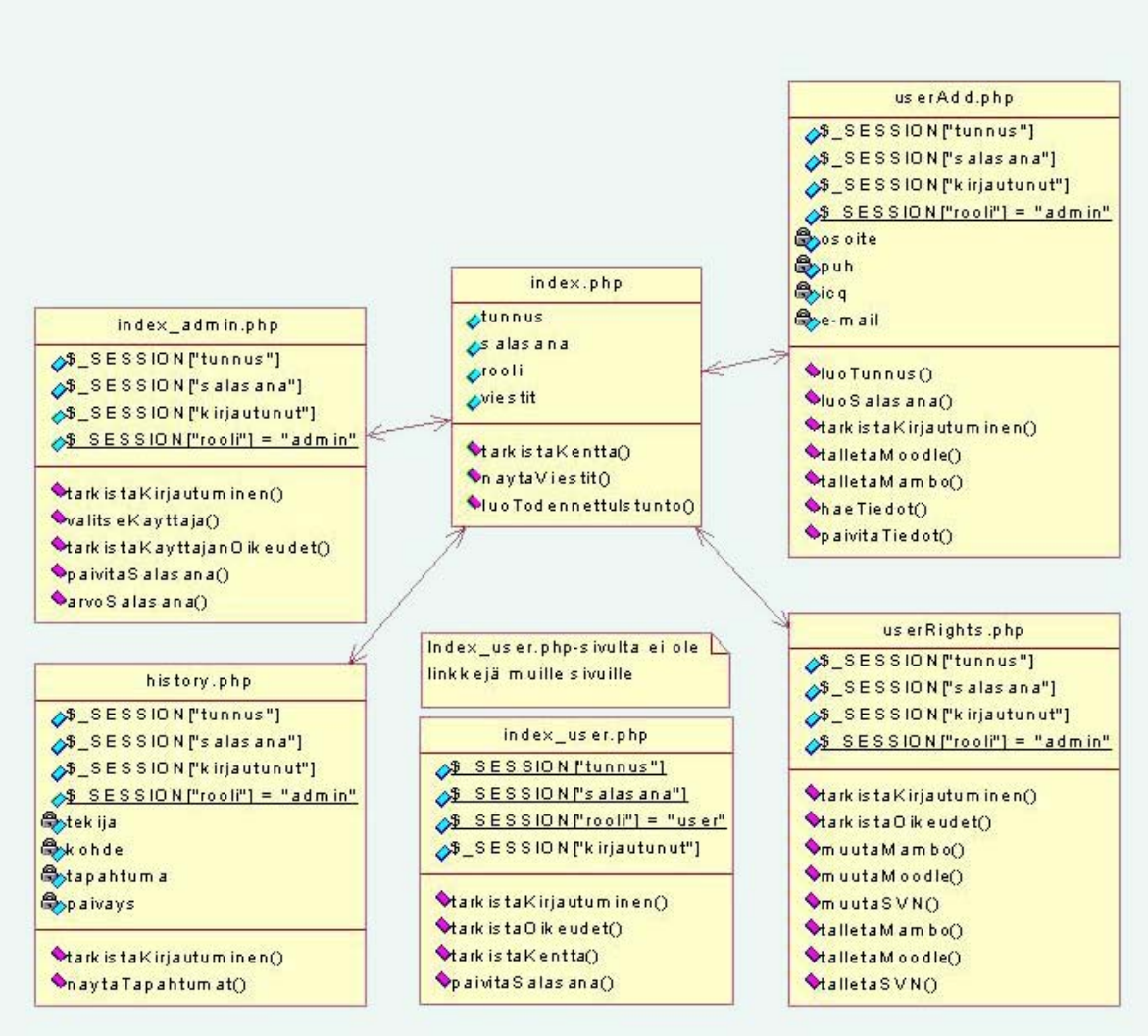
/21/ TietoEnator. [verkkodokumentti]. *Tietoturvallinen ohjelmistojen kehitys.*22.11.2004. [viitattu 3.4.2006]. Saatavissa: http://www.tml.tkk.fi/Opinnot/T-110.470/2004/20041122.pdf

/22/ Suomen Moodle-yhteisö. [verkkodokumentti]. *Moodle-tietous*. [viitattu 6.2.2006]. Saatavissa: http://www.moodle.fi/index.html

/23/ Mambo Foundation Inc. [verkkodokumentti]. *Mambo power in simplicity.* [viitattu 6.2.2006].Saatavissa: http://www.mamboserver.com/

/24/ Collins-Sussman, Ben. [verkkodokumentti]. *Version Control with Subversion.* 2002-2006. [viitattu 22.2.2006] Saatavissa: http://svnbook.red-bean.com/nightly/en/svn-book.html

GateKeeper-ohjelman moduulirakenne



Gatekeeper-ohjelma koostuu moduuleista. Käytännössä jokainen moduuli on erillinen web-sivu. Jokaiselta sivulta on linkki muille ohjelman sivuille.

Index_admin.php-moduulin ohjelmakoodia

```
<?php
/************************************************************
* Sivu: index_admin.php                                     *
*                                                           *
* Toiminnot :                                               *
* 1.Käytäjän roolin tarkastus                               *
* 2.Kirjautumisen tarkastus                                 *
* 3.Luodaan lomake salasanan vaihtamista varten             *
************************************************************/

// Tulostuksen puskuroinnin aloitus:
ob_start();

/**** Tarkistetaan käyttäjän oikea rooli ****/
// roolin pitää olla 1, eli pääkäyttäjälle määrätty rooli.
// Muuten käyttäjä palautetaan kirjautumissivulle

if(($_SESSION["rooli"]!='1'))
        {
        header("Location: index.php?");
        }

/**** Tarkistetaan onko kirjauduttu: ****/
tarkistaKirjautuminen("yes");

/**** HTML-koodi alkaa: ****/
echo "<html>";

/**** Sivun otsikkotiedot: ****/
echo "<head>";
echo "<otsikko>MainPage</otsikko>";

// Evätään pääsy hakukoneilta:
echo "<meta name=\"robots\" content=\"noindex, nofollow\"/>";

// Määritellään käytetty tyylisivu:
echo "<link rel=\"stylesheet\" href=\"template_css.css\" type=\"text/css\"/>";

echo "</head>";

/**** Leipäteksti alkaa: ****/
echo "<body leftmargin=\"0\" topmargin=\"0\" marginwidth=\"0\" marginheight=\"0\">";

/**** Luodaan lomake tietojen syöttöä varten: ****/
echo "<form action=".$_SERVER["PHP_SELF"]." method=\"POST\">";

// Yhdistetään GateKeeper-tietokantaan tietojen hakemista varten:
yhdistaGateKeeper();
if(isset($gatekeeper_yhteys))
{
```

```
// Luodaan taulukko, jossa on kentät tunnusta ja salasanoja varten:
echo "<table border=\"0\">";
// Otsikko:
echo "<tr><td><b>Salasanan muuttaminen:</b></td></tr>";

// Lisätään lomakkeelle tunnus-kenttä
// Tunnukset saadaan tunnus-kenttään SQL-kyselyn avulla:

echo "<tr><td>Tunnus:</td>";
$kysely_kayttajat="select tunnistus_kayttaja from tunnistus order by tunnistus_kayttaja";
$tulos_kayttajat=mysql_query($kysely_kayttajat, $gatekeeper_yhteys)
or die("Ongelma kyselyssä: ".mysql_error());
echo "<td><select name=kayttaja style=\"width:145px\">";
        while ($rivi = mysql_fetch_array($tulos_kayttajat))
        {
        $kayttaja = $rivi['tunnistus_kayttaja'];
        echo "<option value=\"$kayttaja\">$kayttaja";
        }
echo "</select></td>";

// Luodaan lomakkeelle kaksi kenttää uusia salasanoja varten:
echo "<tr><td>Salasana:</td>";
echo "<td><input type=\"password\" name=\"salasana_eka\" maxlength=\"32\"
value=\"\"></td></tr>";
echo "<tr><td>Salasana uudestaan:</td>";
echo "<td><input type=\"password\" name=\"salasana_toka\" maxlength=\"32\"
value=\"\"></td></tr>";

// Lisätään lomakkeelle tallenna- ja tyhjennä-napit:
echo "<tr><td><input type=\"submit\" name=\"tallenna\" value=\"Tallenna\"></td>";
echo "<td><input type=\"submit\" name =\"tyhjennä\" value =\"Tyhjenna\"></td></tr>";

// Päätetään taulukko:
echo "</tr></table>";

// Päätetään lomake:
echo "</form>";

/**** Käsitellään lomakkeella annettut tiedot: ****/

// Tyhjennä ja Tallenna -nappien toiminta:
if ($_POST['tyhjenna'])
        {
        $_POST['tunnus'] = "";
        $_POST["salasana_eka"] ="";
        $_POST["salasana_toka"]="";
        }

// Annettujen tietojen käsittely:
if(isset($_POST["tallenna"]))
        {
        // Salasanojen tarkistus kentanTarkistus -funktiolla:
        kentanTarkistus("ensimmäinen salasana", $_POST["salasana_eka"],
        "string", 4, 32);
```

```
kentanTarkistus("toinen salasana", $_POST["salasana_toka"],
"string", 4, 32);


//Jos salasanojen tarkistuksessa tulee virheitä, näytetään viestit:
if($viestit)
{
naytaViestit();
exit;
}

// Alustetaan muuttujat $kayttaja, $salasana ja $salasana_svn
$kayttaja = $_POST['kayttaja'];
$kayttaja = mysql_real_escape_string($kayttaja);

// Verrataan, ovatko salasanat samat:
// Jos salasanat täsmäävät, salasana tallennetaan salattuna tietokantaan
if (strcasecmp($_POST['salasana_eka'], $_POST['salasana_toka']) == 0)
{
$salasana = $_POST['salasana_eka'];
$salasana = mysql_real_escape_string($salasana);

// Salataan uusi salasana myös svn-tiedostoja varten:
$salasana_svn = crypt($salasana, base64_encode($salasana));

// Luodaan kysely:
$kysely_salasanat ="update tunnistus set tunnistus_salasana =
md5('$salasana'),tunnistus_salasana_svn = '$salasana_svn'
 where tunnistus_kayttaja ='$kayttaja'";

// Suoritetaan kysely:
$tulos_salasanat =mysql_query($kysely_salasanat,
$gatekeeper_yhteys) or die("Ongelma tunnistus-taulun
päivityksessä:".mysql_error());
}

// if (isset (tallenna))-lause loppuu:
}

// if (isset ($mysql-yhteys)) -silmukka päättyy:
}

// Tulostuksen puskuroinnin lopetus:
ob_end_flush();
```

```
/************************************************************
* Ohjelmakoodissa käytetyt funktiot                         *
/************************************************************ /

/************************************************************/
* Funktion Nimi : tarkistaKirjautuminen($tila)              *
*                                                           *
* Tehtävä : Tarkistaa käyttäjän kirjautumisen tilan         *
* $tila-muuttujan avulla.                                   *
************************************************************/

function tarkistaKirjautuminen($tila)
{
switch($tila)
{
case "yes":
        if(!isset($_SESSION["kirjautunut"]))
        {
        header("Location: index.php");
        exit;
        }
        break;

// Mikäli käyttäjä on kirjautunut pääkäyttäjänä ja
// istunto on luotu, käyttäjä ohjataan index_admin.php-sivulle:

case "no":
        if(isset($_SESSION["kirjautunut"]) && $_SESSION["kirjautunut"] === true )
        {
                if(($_SESSION["rooli"]=='1'))
                {
                header("Location:
                index_admin.php?".session_name()."=".session_id());
                }

                if(($_SESSION["rooli"]=='2'))
                {
                header("Location:
                index_user.php?".session_name()."=".session_id());
                }
        }
break;
}

// Mikäli tänne päästää, kirjautuminen on kunnossa
return true;
}
/*****************************************************************************************
* Funktion Nimi : yhdistaGateKeeper()                                                    *
*                                                                                        *
* Tehtävä : Luo pysyvän yhteyden GateKeeper-tietokantaan.                                *
* Huom! Muuttujat $dbhost, $dbuser, $dbpass ja $dbname on alustettu jo                   *
*aikaisemmin.                                                                            *
*****************************************************************************************/
```

```
function yhdistaGateKeeper(){
		global $gatekeeper_yhteys, $dbhost, $dbuser, $dbpass, $dbname;
		($gatekeeper_yhteys = mysql_pconnect("$dbhost", "$dbuser", "$dbpass"))
		|| die("GateKeeper tietokanta yhteys ei onnistunut");

		mysql_select_db("$dbname", $gatekeeper_yhteys)
		|| die("Tietokantaa ei löytynyt: $dbname. Ongelma: ".mysql_error() );
}
/*****************************************************************************
* Funktion Nimi : kentanTarkistus($kentan_nimi, $kentan_tieto,      *
* $kentan_tyyppi, $min_pituus="", $max_pituus="",                   *
*                                                                   *
* Tehtävä : Tarkistaa syötteiden oikeellisuuden tilanteen mukaan    *
*****************************************************************************/

function kentanTarkistus($kentan_nimi, $kentan_tieto,
$kentan_tyyppi, $min_pituus="", $max_pituus=""){

// Taulukko virheitä varten.
global $viestit;

// Hash-taulukko tietotyyppejä" varten.
$tieto_tyypit=array(
		"email"=>$email_regexp,
		"digit"=>"^[0-9]$",
		"number"=>"^[0-9]+$",
		"numero"=>"^[+-]?[0-9\-]+$",
		"alpha"=>"^[a-zA-Z]+$",
		"alpha_space"=>"^[a-zA-Z ]+$",
		"alphanumeric"=>"^[a-zA-Z0-9]+$",
		"alphanumeric_space"=>"^[a-zA-Z0-9 ]+$",
		"string"=>""
);

//Jos kenttä tyyppi on string, ei tarvita tarkistusta regexp:
if ($kentta_tyyppi == "string")
		{
		$kentta_ok = true;
		}
else
		{
		$kentta_ok = ereg($tieto_tyypit[$kentan_tyyppi], $kentan_tieto);
		}

// Jos kenttä on virheellinen, näytetään viesti:
if (!$kentta_ok)
		{
		$viestit[] = "Syötä $kentan_nimi oikein!";
		return;
		}
```

```
// Tarkistetaan syötteen minimi pituus:
if ($kentta_ok && ($min_pituus > 0))
		{
		if (strlen($kentan_tieto) < $min_pituus)
			{
			$viestit[] = "$kentan_nimi on väärin, sen pitäisi olla vähintään
			$min_pituus merkkiä.";
			return;
			}
		}
// Tarkistetaan syötteen maksimipituus:
if ($kentta_ok && ($max_pituus > 0))
		{
		if (strlen($kentan_tieto) > $max_pituus)
			{
			$viestit[] = "$kentan_nimi on väärin , sen pitäisi olla vähemmän
			kuin $max_pituus merkkiä.";
			return;
			}
		}
}// Funktion kentanTarkistus() loppu
/*****************************************************************************
* Funktion Nimi : naytaViestit()                                             *
*                                                                            *
* Tehtävä : Tulostaa viestit tilanteen mukaan                                *
*****************************************************************************/
function naytaViestit() {
global $viestit;
global $otsikko;
if($viestit)
		{
		naytaVirheet($viestit);
		}
}
/*****************************************************************************
* Funktion Nimi : naytaVirheet()                                             *
* Tehtävä : // Funktio luo taulukon havaituista virheistä                    *
* ja tulostaa sen HTML:n avulla                                              *
*****************************************************************************/
function naytaVirheet($viestit) {
print("<b>Ilmoituksia:</b>\n<ul>\n");
foreach($viestit as $viesti){
		echo "<li>$viesti</li>\n";
		}
echo "</ul>\n";
}
/*****************************************************************************
* Ohjelmassa käytettyjen funktioiden esittely loppuu.                        *
*****************************************************************************/
/**** Leipäteksti loppuu: ****/
echo "</body>";
/**** HTML-koodi loppuu: ****/
echo "</html>";
?>
```